DOCUMENT RESUME

ED 088 473 IR 000 306

AUTHOR	Murai, Shinnichi
TITLE	GUIDE-0: An Experimental Information System.
INSTITUTION	Illinois Univ., Urbana. Dept. of Computer Science.
SPONS AGENCY	National Science Foundation, Washington, D.C.
REPORT NO	UIUCDCS-R-73-587
PUB DATE	Aug 73
NOTE	103p.; Master of Science Thesis submitted to the Graduate College of the University of Illinois
EDRS PRICE	MF-$0.75 HC-$5.40
DESCRIPTORS	*Computer Assisted Instruction; Computer Oriented Programs; *Computer Science Education; Data Bases; Higher Education; *Information Retrieval; *Information Systems; Interaction; Man Machine Systems; Masters Theses
IDENTIFIERS	*GUIDE 0; PLATO IV

ABSTRACT

A description is provided of GUIDE-0, an experimental information system. The system serves as a bibliographic aid for students who are taking introductory computer science courses whose material is at least partially implemented via PLATO-IV lessons. Following a brief introduction to the system in Chapter I, the second Chapter describes the functions of GUIDE-0, including the search for and display of lessons, display lesson descriptors, display course outline and display course student record. Chapter III treats the system's data base, dealing with its lesson catalog, course record, and the implementation of the data base. Chapter IV covers GUIDE-0's modules, describing its 1) instruction decoder and controllers, 2) lexical analyzer for search prescription, 3) syntax and semantic analyzer of search prescription, 4) search range vector calculator, 5) sequential search module, 6) binary search module, 7) message editors, and 8) miscellaneous modules. The final chapter provides a functional specification of the data base editor, reviewing in turn the lesson catalog and course record editors. (PB)

ED 088473

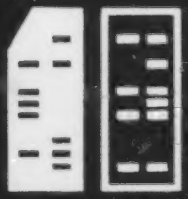
UIUCDCS-R-73-587

GUIDE-O
-- AN EXPERIMENTAL INFORMATION SYSTEM --

by

Shinnichi Murai

August, 1973



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

GUIDE-O
- AN EXPERIMENTAL INFORMATION SYSTEM --

BY

SHINNICHI MURAI

B. Eng., Kyoto University, 1963

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1973

Urbana, Illinois

U.S. DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION
THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

ACKNOWLEDGMENT

The author wishes to thank Professor Jurg Nievergelt for providing me with the opportunity to work on the project and for many valuable suggestions and encouragements on the work. Thanks are also due to Mr. Dave Eland for suggestions and corrections on the manuscript and Messrs. Jean Pradels and Mateti Prabhaker for discussions on the project. Special thanks go to Mrs. Connie Slovak and June Wingler for the quick and excellent typing and preparation of the paper.

TABLE OF CONTENTS

		Page
I.	INTRODUCTION	1
II.	THE FUNCTIONS OF GUIDE-O	3
	1. Search for and Display Lessons Which Match a Given Search Prescription	3
	1.1 Input (Search Prescription)	3
	1.2 Output (Lesson Names and Abstracts)	3
	1.3 Notes	4
	2. Display Lesson Descriptors	4
	3. Display Course Outline	6
	4. Display Student Record	6
III.	THE DATA BASE OF GUIDE-O	7
	1. Lesson Catalog	7
	1.1 Lesson Catalog 1	10
	1.2 Lesson Catalog 2	11
	1.3 Keyword Table	12
	2. Course Record	13
	2.1 Course Outline	13
	2.2 Student Record	14
	2.3 Student Directory	15
	2.4 Course Directory	16
	3. Implementation of the Data Base	18
	3.1 PLATO IV Storage Organization	18

3.2 Implementation of the Data Base in Common Storage..... 20

IV. THE MODULES OF GUIDE-0..... 22

1. Instruction Decoder and Controllers..... 22

1.1 Instruction Decoder "idecode"..... 23

1.2 Controller 1 (Search for Lessons Which Match Search Prescription) "idclsp"..... 23

1.3 Controller 2 (Display Lesson Descriptors) "idcdld"..... 26

1.4 Controller 3 (Display Course Outline) "idcdco"..... 26

1.5 Controller 4 (Display Student Record) "idcdsr"..... 26

2. Lexical Analyzer for Search Prescription ("lexi")..... 31

2.1 General..... 31

2.2 Input..... 31

2.3 Output..... 32

2.4 Functional Description..... 32

3. Syntax and Semantics Analyxer of Search Prescription ("parser")..... 35

3.1 General..... 35

3.2 The Grammar of the Search Prescription..... 35

3.3 Precedence Relations Between Operators..... 35

3.4 Input..... 36

3.5 Output..... 36

3.6 Functional Description..... 36

4. Search Range Vector Calculator "calcsrv"..... 37

4.1 Search Range Vector..... 37

4.2 Basic Function of "calcsrv"..... 38

4.3 Input..... 38

4.4 Output..... 38

4.5 Functional Description of "calcsrv"..... 38

5. Sequential Search Module "ssearch"..... 40

5.1 General..... 40

5.2 Input (Parameters)..... 40

5.3 Output..... 41

5.4 Functional Description..... 42

6. Binary Search Module "bsearch"..... 42

6.1 General..... 42

6.2 Input (Parameters)..... 42

6.3 Output..... 43

6.4 Functional Description..... 43

7. Message Editors..... 43

7.1 Message Editor 1 "edtlsp"..... 44

7.2 Message Editor 2 "edtdld"..... 44

7.3 Message Editor 3 "edtdco"..... 46

7.4 Message Editor 4 "edtdsr"..... 46

7.5 Subeditor "edtdata"..... 49

8. Miscellaneous Modules..... 49

8.1 "inprecd" -- Initialize the Precedence Matrix..... 49

8.2 Setting the Experimental Data into the Data Base..... 51

8.3 Displaying the Variables for Debugging Purpose..... 51

V. FUNCTIONAL SPECIFICATION OF THE DATA BASE EDITOR..... 53

1. Lesson Catalog Editor..... 53

1.1 Addition of a New Lesson Name into Lesson Catalog..... 53

1.2 Replacement of Abstract..... 54

1.3 Addition of a Subject Category Code..... 54

1.4 Deletion of a Subject Category Code..... 54

1.5 Replacement of a Subject Category Code..... 54

1.6 Addition of a Keyword..... 54

1.7 Deletion of a Keyword..... 55

1.8 Replacement of a Keyword..... 56

1.9 Addition of a Relation to Other Lesson..... 56

1.10 Deletion of a Relation to Other Lesson..... 56

1.11 Replacement of a Relation to Other Lesson..... 57

1.12 Replacement of Lesson Type..... 57

1.13 Replacement of the Time Required..... 57

2. Course Record Editor..... 57

2.1 Registration of a Course..... 57

2.2 Deletion of a Course..... 58

2.3 Insertion of a Lesson into Course Outline..... 58

2.4 Deletion of a Lesson from Course Outline..... 58

2.5 Replacement of a Lesson in Course Outline..... 59

2.6 Registration of a Student into a Course..... 59

2.7 Deletion of a Student from a Course..... 59

2.8 Modification of "Date" and "Performance" of Student Record..... 60

2.9 Garbage Collection..... 60

LIST OF REFERENCES..... 62

APPENDIX

LISTING OF GUIDE-0..... 63

LIST OF TABLES

Table Page

1. Subject Category..... 5
2. The Implementation of GUIDE-0 Data Base.....21
3. Code of Token.....33
4. Precedence Matrix.....36
5. Data Base Initialization Modules.....51

LIST OF FIGURES

Figure		Page
3.1	File Structure of Lesson Catalog	8
3.2	File Structure of Course Record	9
3.3	Storage Hierarchy of PLATO IV	19
4.1	Main Flow of GUIDE-0	24
4.2	Flow of Controller for Function 1	25
4.3	Structural Relationship between Modules - 1	25
4.4	Flow of Controller 2	27
4.5	Structural Relationship between Modules - 2	27
4.6	Flow of Controller 3	28
4.7	Structural Relationship between Modules - 3	28
4.8	Flow of Controller 4	29
4.9	Structural Relationship between Modules - 4	30
4.10	State Diagram of Lexical Analyser	33
4.11	Output Format for the Function 1	45
4.12	Output Format for the Function 2	47
4.13	Output Format for the Function 3	48
4.14	Output Format for the Function 4	50

I. INTRODUCTION

The project to automate a portion of the introductory computer science courses onto the PLATO IV system [2] has been initiated in order to meet the need to offer those courses to a sharply growing number of people whose background and ability differ considerably [1]. The automation of instruction has two aspects, quantitative and qualitative. The automation of grading exams can be considered as an example of quantitative effect. Many lessons in PLATO IV can be said to have qualitative effect in the sense that they allow personal and interactive communication with each student. The main objective of GUIDE [1, 3], a conversational information system, is to help students in selecting PLATO IV lessons to study. This selection is made in an interactive manner, and is based on the student's past activity and performance. The necessity of this kind of system is apparent, considering the rapid growth of the number of students with different background and ability, and also the increasing number of lessons in the PLATO IV system. GUIDE-0 is intended to serve as an experimental system for GUIDE to investigate the formation of search prescriptions, keyword repertoire, data items to be included in the data base, structure of the data base, search algorithms, etc. in the PLATO IV environment. It is also intended to fulfill the practical necessity for GUIDE until it is actually implemented. The capabilities of GUIDE-0 are "reduced" in the following sense:

(a) The language for the communication between GUIDE-0 and its users is severely restricted. GUIDE will allow communication by natural language (English),

but GUIDE-0 is restricted to its own special language (instructions).

(b) No judging function based on each student's past record is included in GUIDE-0. It displays only the information in the data base. For example, students can't ask GUIDE-0 such questions as "What should I study today?". A student is expected to ask GUIDE-0 to display his past record and/or course outline, and to judge by himself what to study.

(c) GUIDE-0 is not "clever" enough. It searches the data base based upon exactly the user specified terms. For example, suppose a user asked to search the lessons which deal with "2-dimensional arrays", and also suppose that not "2-dimensional arrays" but "2-dimensional array" is included in the keyword repertoire. Then, GUIDE-0 can't search by the plural form. Also, GUIDE-0 won't automatically do the expansion of a search prescription to a broader term, for example, from "2-dimensional array" to "array", although GUIDE-0 suggests users to do so, when it can't find what users asked.

(d) The scope of the data base is small, and the structure of the data base is influenced by the scope and current status of the ever expanding PLATO IV system architecture.

II. THE FUNCTIONS OF GUIDE-O

1. Search for and Display Lessons Which Match a Given Search Prescription

Users are supposed to give GUIDE-O a search prescription, i.e., a logical expression of keywords which represents their interest. Then GUIDE-O searches the data base for the lessons which match the given search prescription, and displays the names and abstracts of those lessons.

1.1 Input (Search Prescription)

The input for this function is a logical expression of keywords followed by a semicolon. The allowed logic operations in the expressions are AND(*), OR(+) and NOT('). Operators and operands may be separated by blanks, and parantheses (nested to any level) are allowed.

For example, suppose that a user wants to study about data structures other than arrays, in any languages other than PL/1 and ALGOL. Then the search prescription for him could be as follows:

```
data structure * array'*(pl/1 + algol)';
```

1.2 Output (Lesson Names and Abstracts)

The output of this function is a list of lesson names and abstracts which match the given search prescription. The following is an example:

LESSON	ABSTRACT
racetrack	Simulation Experiment
somaga	Software Management Game to teach Programming
montecarlo	Area Calculation by Monte Carlo Method

If the number of lessons to be displayed exceeds the display size, the lessons are paged and the first page is displayed at first. "NEXT" and "BACK" keys are used to turn pages forward or backward.

1.3 Notes

Each keyword in the search prescription must be exactly the one listed in the Keyword Table of GUIDE-O. As explained in the introduction, for example, if only the singular form of a keyword is listed and if a user specifies the plural form, GUIDE-O does not accept the search prescription and tells the user that the keyword is not included in the repertoire.

If GUIDE-O cannot find any lessons which match the search prescription given, GUIDE-O suggests that the user search by a broader term.

If there are syntactic errors in a search prescription, GUIDE-O asks the user to correct it.

2. Display Lesson Descriptors

The user is asked to give a lesson name. Then GUIDE-O searches the data base for the specified lesson, and displays its lesson descriptors. The lesson descriptors consist of the following items:

(a) Lesson name -- the name of the lesson which is registered in PLATO IV system.

(b) Type -- the type of the lesson such as "practice", "examination", etc.

(c) Abstract - a brief explanation about the contents of the lesson.

(d) Subject category -- each lesson included in the data base of GUIDE-O is classified into one or more of the categories listed in Table 1.

BEST COPY AVAILABLE

0. GENERAL

1. INTRODUCTION
   - 1.0 General
   - 1.1 Programming
   - 1.2 Computers
   - 1.3 Applications

2. PROGRAMMING LANGUAGES
   - 2.0 General
   - 2.1 PL/I
   - 2.2 Fortran
   - 2.3 Basic

3. PROGRAMMING LANGUAGE CONCEPTS
   - 3.1 Introductory
     - 3.11 General Information
       - 3.111 Operating Systems -- Hasp, Express
       - 3.112 Compilers -- PL/I, PL/C
       - 3.113 Job Control Language
     - 3.12 Overview of a language
       - 3.121 General Program Format -- statements, labels, comments, line continuation, character sets, delimiters, separators, use of blanks, punctuation
       - 3.122 Program Structure -- sequencing rules, blocks, groups
       - 3.123 Basic Instruction Types -- executable unit
       - 3.124 Identifiers, Keywords, Reserved Words
       - 3.125 Interaction with Operating System and Environment
       - 3.126 Special Conventions, Features
       - 3.127 Illustrative Programs
   - 3.2 Data Types
     - 3.21 Arithmetic -- constants, variables, fixed, float, precision
     - 3.22 Character String
     - 3.23 Bit String
     - 3.24 Pointer
     - 3.25 Labels
     - 3.26 Declarations, Attributes
   - 3.3 Data Operations
     - 3.31 Arithmetic
       - 3.311 Operators
       - 3.312 Assignment Statement
       - 3.313 Expressions
       - 3.314 Initialization
       - 3.315 Built-in Functions
     - 3.32 String
       - 3.321 Operators
       - 3.322 Built-in Functions
     - 3.33 Comparison
       - 3.331 Operators
       - 3.332 Expressions
     - 3.34 Logical
       - 3.341 Operators
       - 3.342 Expressions
   - 3.4 Data Structures
     - 3.41 Arrays
     - 3.42 Hierarchical Structures
     - 3.43 List
   - 3.5 Data Storage
     - 3.51 Storage Types -- static, external, automatic, based, . . .
     - 3.52 Dynamic Storage Allocation
   - 3.6 Control Statements
     - 3.61 Unconditional Branch -- goto
     - 3.62 Conditional Branch -- computed goto, if, then, else
     - 3.63 Loops, Iteration -- do, while
     - 3.64 Recursion
   - 3.7 Input/Output
     - 3.71 Data formats -- (F, E, A, . . .)
     - 3.72 Printer carriage control -- page, line, column
     - 3.73 Types of I/O
       - 3.731 Stream
       - 3.732 Record
       - 3.733 Internal -- (get string, put string)
   - 3.8 Subprograms
     - 3.81 Functions
       - 3.811 Built-in -- arithmetic, string
       - 3.812 Library
       - 3.813 Programmer Defined -- internal, external
       - 3.814 Generic
       - 3.815 Recursive
     - 3.82 Subroutines
   - 3.9 Others
     - 3.91 Interrupts
     - 3.92 Debugging aids
     - 3.93 Compiler Directives, Macros

4. APPLICATIONS
   - 4.0 General
   - 4.1 Natural Sciences
   - 4.2 Engineering
   - 4.3 Social Sciences
   - 4.4 Humanities
   - 4.5 Business Data Processing
   - 4.6 Education

5. TECHNIQUES
   - 5.0 General
   - 5.1 Numerical Methods
     - 5.10 General
     - 5.11 Error Analysis; Computer Arithmetic
     - 5.12 Function Evaluation
     - 5.13 Interpolation; Functional Approximation
     - 5.14 Linear Algebra
     - 5.15 Nonlinear and Functional Equations
     - 5.16 Numerical Integration and Differentiation
     - 5.17 Differential Equations
     - 5.18 Integral Equations
   - 5.2 Combinatorial
     - 5.20 General
     - 5.21 Enumeration
     - 5.22 Sorting
     - 5.23 Searching
     - 5.24 Graphs
   - 5.3 Optimization
   - 5.4 Simulation
     - 5.40 General
     - 5.41 Discrete
     - 5.42 Continuous
     - 5.43 Monte Carlo
   - 5.5 Heuristic
     - 5.50 General
     - 5.51 Heuristic Search
     - 5.52 Adaptive Programs
     - 5.53 Pattern Recognition
   - 5.6 Programming Techniques
     - 5.61 List Processing

Table 1. Subject Category

(e) Keywords -- keywords which represent the contents of the lesson.

(f) Time required -- the estimated time required to go through the lesson.

(g) Relations to other lessons -- the relations between lessons such as "prerequisite of the lesson", "sequel to the lesson", etc.

3. Display Course Outline

The user specifies the course and section number. Then GUIDE-0 displays the course outline of the specified course. The Course Outline is a list of lesson names with the dates by which each lesson must be taken, the estimated time required to go through the lesson, and the expected performance in the lesson (if it is a "practice", "exercise" or "exam" type lesson).

4. Display Student Record

The user specifies the course and section number, his name and his social security number. Then GUIDE-0 displays his record in the specified course. The student record is a list of lesson names with the last date the student took each lesson, the time the student spent on the lesson, and his performances in the lesson (if it is a "practice", "exercise" or "exam" type lesson).

III. THE DATA BASE OF GUIDE-O

The data base for GUIDE-O consists of two parts, the Lesson Catalog and Course Record, each of which consists of several files. The Lesson Catalog contains the information related to lessons such as the abstracts of the lessons, the keywords attached to the lessons, etc. and is mainly used for functions 1. and 2. of the preceding chapter. The Course Record contains the information related to course activity such as course outlines, each student's performance in each lesson, etc. and is used for functions 3. and 4. The Lesson Catalog consists of three files: Lesson Catalog 1, Lesson Catalog 2 and Keyword Table. The Course Record consists of four files: Course Directory, Course Outline, Student Directory and Student Records. All of these files are stored in the "common" storage provided by PLATO IV [4]. Figures 3.1 and 3.2 illustrate the structure of the Lesson Catalog and Course Record.

1. Lesson Catalog

The Lesson Catalog provides the information necessary to know what a lesson is about, or to retrieve the lessons which are supposed to be related to a user's interest. The Lesson Catalog consists of three files: Lesson Catalog 1, Lesson Catalog 2 and Keyword Table. The first two are related to what a lesson is about, and the last is used for retrieval purposes. The difference between Lesson Catalog 1 and Lesson Catalog 2 is as follows:

(a) Lesson Catalog 2 is sorted by lesson name, thus allowing a binary search by lesson name; on the

Lesson Catalog 1

Lesson Name	Abstract	Subject Category	Keywords	Not Used
montecarlo	Area Calculation by Monte-Carlo Method	5230 1300	$\underbrace{K_1}_{4b}$ K_2 ... K_7	
10 chars	62 characters	2x4 chars	7x8 bits	4 bits
1 word	7 words		1 word	

9 words

Lesson Catalog 2

Lesson Name	No.	Time Required	Relations to Other Lessons	Not Used	Type
files	14	t_1	$\underbrace{\rho_1}_{4b}\underbrace{l_1}_{6b}\rho_1 l_2 \rho_2 l_3 \cdots$		y_1
montecarlo	1	t_2	4b 6b		y_2
pll data	17	t_3			y_3
10 chars	6 bits	6 bits	4x(4+6) bits	4 bits	4 bits
1 word			1 word		

2 words

Keyword Table

Keyword	Retrieval Code
array	1010.....011
assignment statement	0100.....1000
20 characters	60 bits
2 words	1 word

3 words

Figure 3.1 File Structure of Lesson Catalog

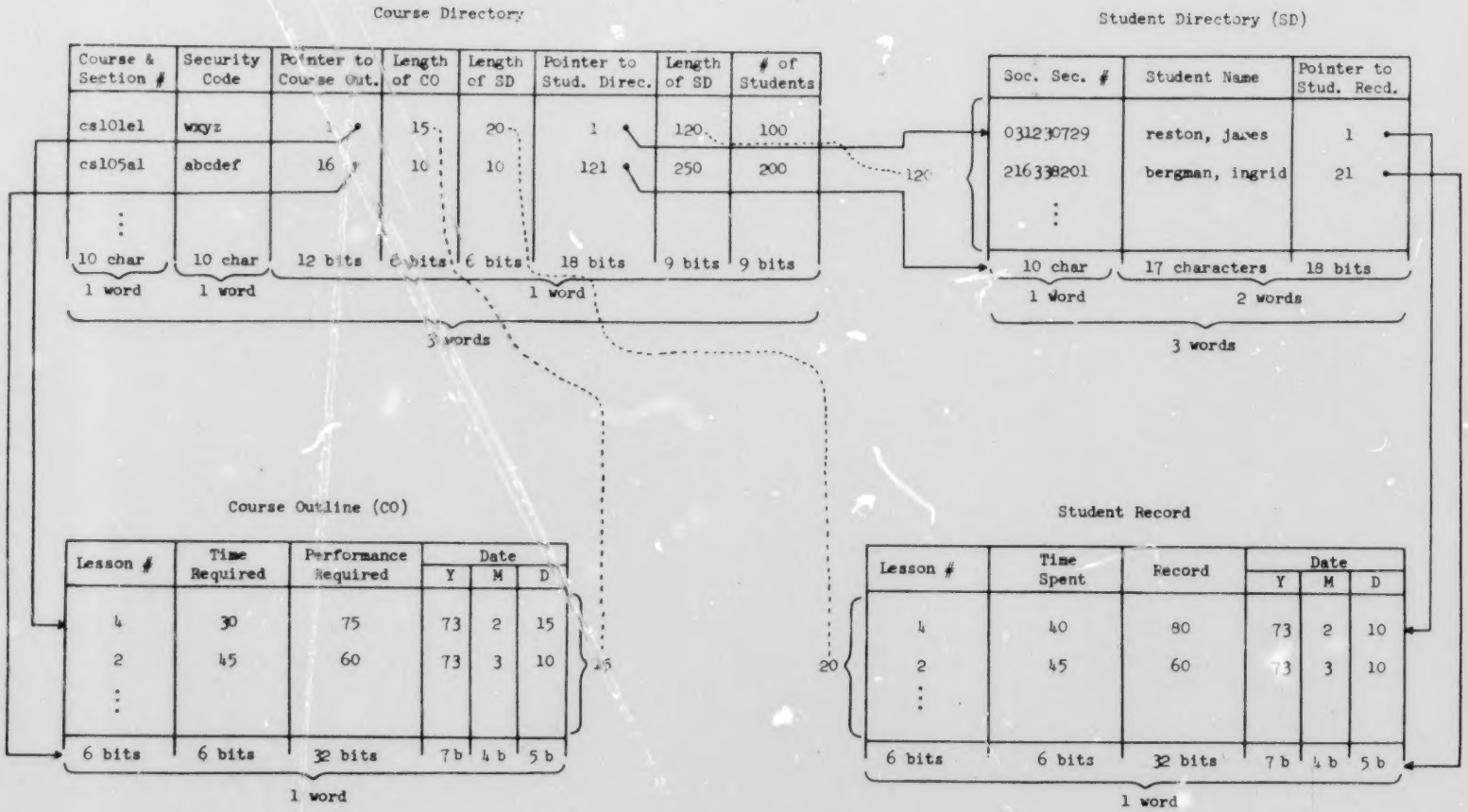


Figure 3.2 File Structure of Course Record

other hand, Lesson Catalog 1 is sorted by lesson number. (Actually no sorting operation is done on Lesson Catalog 1. The location of a lesson in Lesson Catalog 1 is the lesson number of the lesson.)

(b) Lesson Catalog 2 contains different information about the same lessons which are stored in Lesson Catalog 1.*

At the current implementation, the information for up to 60 lessons can be stored in the Lesson Catalogs and up to 256 keywords can be stored in the Keyword Table.

Figure 3.1 illustrates the structure of the Lesson Catalog.

1.1 Lesson Catalog 1

Lesson Catalog 1 consists of the following four fields:

(a) Lesson Name (lessonml) -- 10 characters

This field contains the lesson name of maximum 10 characters.

(b) Abstract (abstrct) -- 62 characters

This field contains the very brief explanation of the lesson of maximum 62 characters.

(c) Subject Category -- 2x4 characters

This field contains up to two codes each of which consists of 4 digits (characters) and represents the category of the lesson.

---

*Originally, Lesson Catalog 2 was intended to provide only an index to Lesson Catalog 1. However, since the unit of information processing in the PLATO IV system (in the TUTOR language) is basically a word (not a character or a byte) and the available storage area is severely restricted, it was decided to store a part of the information about lessons into the area of Lesson Catalog 2 which would otherwise be wasted [4, 5].

(d) Keyword -- 7x8 bits

This field contains up to seven keyword identity codes (the location of the keyword in Keyword Table) of 8 bits. Thus a maximum of 8 keywords can be attached to each lesson.

Thus Lesson Catalog 1 consumes 9 words of memory per lesson, i.e., 640 words are necessary for 60 lessons.

1.2 Lesson Catalog 2

Lesson Catalog 2 consists of the following five fields:

(a) Lesson Name (lessnm2) -- 10 characters

This field contains the lesson name of up to 10 characters.

(b) Lesson Number (glessnn) -- 6 bits

This field contains the location of the lesson in Lesson Catalog 1 (the lesson number).

(c) Time Required (gtime) -- 6 bits

This field contains the average time required to finish the lesson.

(d) Relations to Other Lessons -- 4x(4+6) bits

This field contains up to 4 relation identity codes of 4 bits each of which is followed by the lesson number (6 bits) of the lesson which has the relationship specified by the relation identity code with the lesson specified in the Lesson Name field. For example, if the lesson has two prerequisites 12 and 13, and one sequel 14; the Relation field should look like

$\underbrace{\rho_1}_{4\text{ bits}}\ \underbrace{12}_{6\text{ bits}}\ \underbrace{\rho_1\ 13\ \rho_2\ 14}_{\cdots\cdots\cdots\cdots}\ \underbrace{000\ldots0}_{10\text{ bits}}$

where ρ_1, ρ_2 are the relation identity codes for "prerequisite" and "sequel" respectively, and 11, 12, 13 are lesson numbers.

(e) Type (gtype) -- 4 bits

This field contains the lesson type identity code of 4 bits.

Lesson Catalog 2 consumes 2 words per lesson, thus 120 words per 60 lessons.

1.3 Keyword Table

The Keyword Table consists of the following two fields:

(a) Keyword (keyword) -- 20 characters

This field contains a keyword (or phrase) of up to 20 characters.

(b) Retrieval Code (retcode) -- 60 bits

This field contains a retrieval code of 60 bits which is attached to the keyword specified in the Keyword field. Each bit of a retrieval code represents a lesson and the bit position corresponds to the lesson number. For example, consider the following case:

Keyword	Retrieval Code
program flow control	00100001100.....0

This indicates that the keyword "program flow control" is attached to three lessons, the lesson numbers of which are 3, 8 and 9 respectively. Thus if the user specifies the keywords which represent

his interest, then the GUIDE-O program searches the Keyword Table, obtains the retrieval codes, and gives the necessary information (lesson names and abstracts) about the lessons which appear in the retrieval codes.

The Keyword Table is sorted alphabetically by keywords, allowing a binary search for a given keyword. The Keyword Table consumes 3 words of memory per keyword, thus 768 words per 256 keywords.

2. Course Record

The Course Record provides students and instructors the information such as course outlines, students' performances in some lessons, the date when students took lessons, etc. The Course Record consists of the following four files:

(a) Course Directory,

(b) Course Outline

(c) Student Directory,

(d) Student Record.

The Course Directory contains the pointers to course outlines and to student directories, and some other administrative information. The Course Outline contains course outlines (the schedule of lessons to be taken in each course). The Student Directory contains lists of students enrolled in the courses and pointers to each student's own record. The Student Record contains students' records such as performance in lessons, time spent in lessons, etc.

2.1 Course Outline

The Course Outline is the schedule of lessons which are to be taken by students who are enrolled in the course. It consists of the following five fields:

(a) Lesson Number -- 6 bits

This field contains a lesson number (the location of the lesson in Lesson Catalog 1).

(b) Time Required -- 6 bits

This field contains the average time or the maximum time to finish the lesson specified in the Lesson Number field.

(c) Performance Expected -- 32 bits

This field contains the performance expected of the lesson specified in the Lesson Number field.

(d) Date -- 16 bits

This field contains the date by which students are expected to finish the lesson specified in the Lesson Number field. The first 7 bits contain the year, the next 4 bits the month, and the last 5 bits the day.

Thus, the Course Outline occupies 2 words of memory per lesson.

2.2 Student Record

The Student Record stores the various student records such as the performance in a lesson, the last date the student took the lesson, etc. for all students who are enrolled in the courses listed in Course Directory. The Student Record has exactly the same format as the Course Outline, consisting of the following 4 fields:

(a) Lesson Number -- 6 bits

This field contains a lesson number (the location of the lesson in Lesson Catalog 1).

(b) Time Spent -- 6 bits

This field contains the time spent by a student to finish the lesson specified in the Lesson Number field.

(c) Records -- 32 bits

This field contains coded records of the student's performance in the lesson specified in the Lesson Number field.

(d) Date -- 16 bits

This field contains the last date the student took the lesson specified in the Lesson Number field.

The Student Records occupy 2 words of memory per lesson (same as Course Outline).

2.3 Student Directory

The Student Directory contains lists of students who are enrolled in the courses listed in the Course Directory, and the pointers to each student's student record. Each course has its own student directory and is sorted by the social security number of the students who are enrolled in the course. If the same student takes two different courses, his name appears twice in two student directories. The Student Directory consists of the following three fields:

(a) Social Security Number -- 9+1 digits (characters)

This field contains the social security number of a student as a character string of 9 digits (characters) +1 blank character.

(b) Student Name -- 17 characters

This field contains a student's name of up to 17 characters.

(c) Pointer to Student Record -- 18 bits

This field contains a pointer to the student record of the student specified in Social Security Number and

Student Name field. The logical location or array subscript, not the physical address, is meant by "pointer".

Thus, the Student Directory occupies 3 words of memory per student.

2.4 Course Directory

The Course Directory contains various administrative data, consisting of the following eight fields:

(a) Course and Section Number (coursen) -- 10 characters

This field contains course and section number (e.g. cs101e1, math105a1, etc.).

(b) Security Code (seccode) -- 10 characters

This field contains security code of up to 10 characters, which protects the privacy of student records.

(c) Pointer to Course Outline (pcoutln) -- 12 bits

This field contains a pointer to the course outline of the course specified by Course and Section Number field. The logical location or array subscript, not the physical address, is meant by "pointer".

(d) Length of Course Outline (lcoutln) -- 6 bits

This field contains the length of the course outline, i.e., the number of lessons contained in the course outline for the course specified in Course and Section Number field.

(e) Length of Student Records (lsrecrd) -- 6 bits

This field contains the length of student records, i.e., the maximum number of lessons to be recorded in each student record. The number contained in this field should be equal to or greater than that of Length of Course Outline. If both numbers are equal, only the lessons listed in the Course Outline are recorded in the Student Record.

(f) Pointer to Student Directory -- 18 bits

This field contains a pointer to the student directory of the course specified in Course and Section Number field.

(g) Length of the Student Directory -- 9 bits

This field contains the length of the student directory of the course specified in Course and Section Number field, i.e., the maximum number of students to be enrolled in the course.

Note that this number is used to reserve a space for the student directory at the beginning of a semester. No more students than this number can be registered in the course under the current version of the GUIDE-0 editor.

(h) Number of Students -- 9 bits

This field contains the current number of students who are registered in the course specified in Course and Section Number field.

Thus, the Course Directory occupies 3 words of memory per course.

3. Implementation of the Data Base

3.1 PLATO IV Storage Organization [4, 5]

The Storage hierarchy of the PLATO IV system consists of three levels (see Figure 3.3)

(a) Central Memory (CM),

(b) Extended Core Storage (ECS),

(c) Disk.

Central Memory is used for the lesson which is currently being executed by the CPU, and so the contents of central memory stays the same only within a single time slice. Extended Core Storage is used to store the lessons which are being "taken" by students currently sitting at a terminal. For example, if 20 students are taking 5 different lessons, those 5 lessons are stored in ECS at the same time. The Disk is used as a permanent storage device for all lessons in the PLATO IV system.

Each lesson in PLATO IV has access to two kinds of variables, "student variables" and "common variables". A set of 150 student variables is attached to each student and is stored in disk. A set of maximum 4186 common variables is attached to a lesson (if necessary) and is also stored in disk. Whenever a lesson is condensed, the student variables attached to the student who is going to use the lesson, and the common variables attached to the lesson being condensed are transferred into ECS from disk. When a time slice is given to a student by the system program of PLATO IV, the lesson the student is working on, the student variables of the student, and in case of automatic loading mode the first 1500 out of 4186 common variables attached to the lesson (if any) are loaded into the central memory. In case of non-automatic loading mode, the loading of up to 1500 common variables is specified by the instruction in a lesson. In this case you can specify

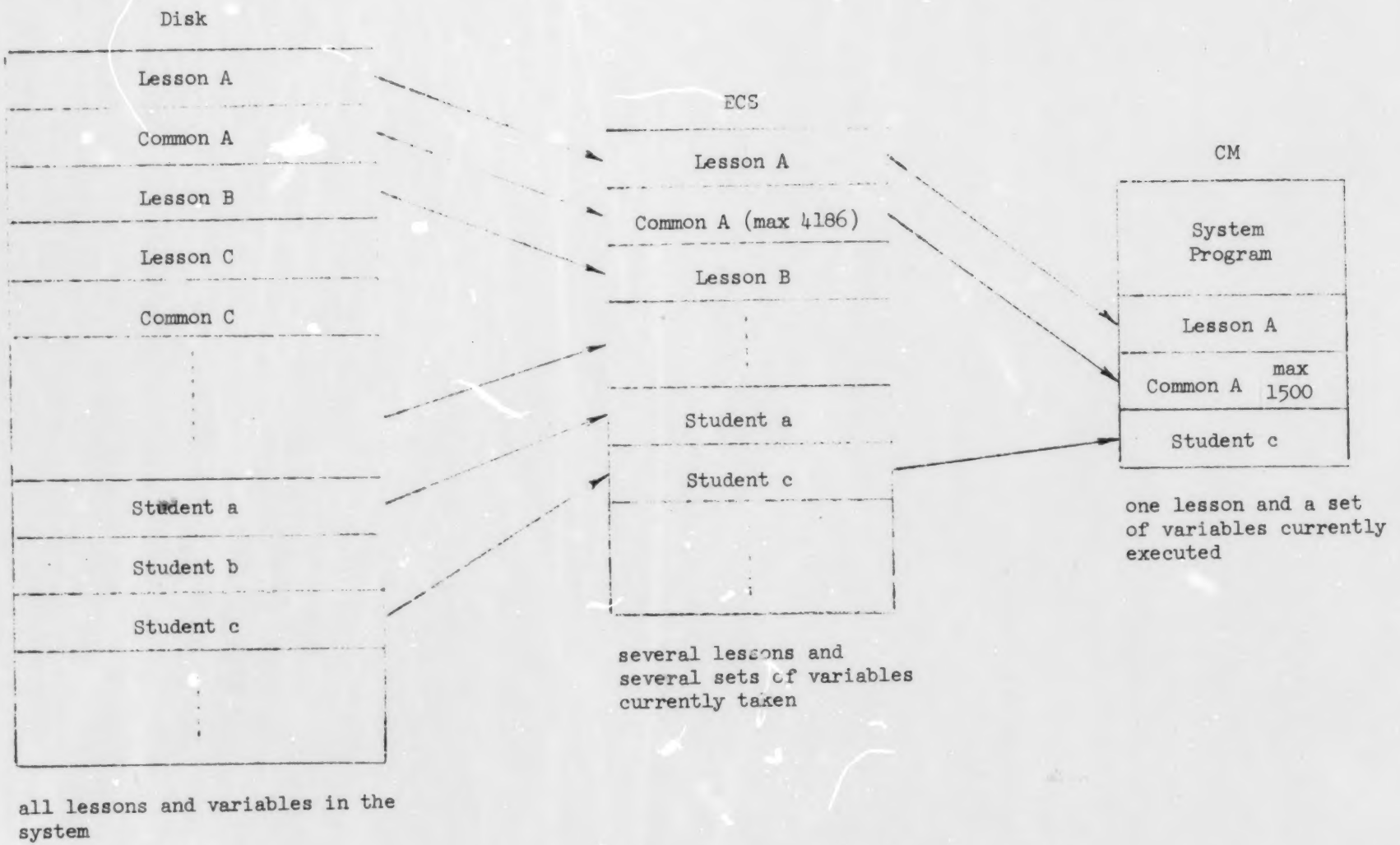


Figure 3.3 Storage Hierarchy of PLATO IV

which part of 4186 common variables to load. In other words, even though 150 student variables and eventually 4186 common variables can be accessed by a lesson, only 150 student variables and 1500 common variables can be accessed at the same time.

3.2 Implementation of the Data Base in Common Storage

The data base of GUIDE-0 is implemented in common storage area. Since the amount of storage area which is available as "common" storage is limited to 4186 words as explained above, the maximum number of items of each file is limited as shown in Table 2. This would be acceptable considering the experimental nature of GUIDE-0. The amount of storage used and the location of each file are also shown in Table 2.

File	Max. No. of Items	Amount of Storage	Location
Lesson Catalog 1	60 lessons	540 (words)	1501 ~ 2040
Lesson Catalog 2	60 lessons	120	2041 ~ 2160
Keyword Table	256 keywords	768	2161 ~ 2928
Course Directory	10 courses	30	1 ~ 30
Course Outline	150 lessons	150	3001 ~ 3150
Student Directory	150 students	450	31 ~ 480
Student Record	1020 lessons	1020	481 ~ 1500

Total = 3078 words

Table 2. The Implementation of GUIDE-0 Data Base

IV. THE MODULES OF GUIDE-O

GUIDE-O consists of the following 8 major modules:

1. Instruction Decoder and Controllers (idecode)
2. Lexical Analyzer of Search Prescription (lexi)
3. Syntax and Semantics Analyzer of Search Prescription (parser)
4. Search Range Vector Calculator (calcsrv)
5. Sequential Search Module (ssearch)
6. Binary Search Module (bsearch)
7. Message Editors
8. Miscellaneous Modules

The modules other than 1 are called (joined) as subroutines by module 1 or the others, and the controllers (parts of 1) are basically sequences of subroutine-calls. The main flow of GUIDE-O and the structural relationship between modules are shown in the following section.

1. Instruction Decoder and Controllers

This module is further subdivided into 5 submodules:

(a) Instruction Decoder

(b) Controller 1 (search for lessons which match search prescription)

(c) Controller 2 (display lesson descriptors)

(d) Controller 3 (display course outline)

(e) Controller 4 (display student record)

The main flow of GUIDE-O can be shown in terms of these 5 sub-modules as in Figure 4.1.

1.1 Instruction Decoder "idecode"

The Instruction Decoder displays the four functions of GUIDE-O described in Chapter 3 to the user and asks him to choose one of them. According to his response, the Instruction Decoder jumps to one of the following four controllers.

1.2 Controller 1 (Search for Lessons Which Match Search Prescription) "idclsp"

As shown in Figure 4.2, Controller 1 receives a search prescription as described in II.1.1. The search prescription is stored in the array "spresc". Then Controller 1 joins (calls) the Syntax and Semantics Analyzer. The Syntax and Semantics Analyzer parses the search prescription stored in "spresc" and puts the result into the array "postfix". The result is a postfix notation with keywords as operands and three kinds of logic operators (AND, OR, NOT) as operators, as seen by the name of the array.

Next, Controller 1 joins the Search Range Vector Calculator. The Search Range Vector Calculator goes through "postfix" and calculates (via the logical operations AND, OR, NOT) the search range vector and puts the resultant vector into a location of the array "tsrange". The location is specified by the first location of the array "pstack", i.e., pstack(1).

After this, Controller 1 joins the Message Editor 1. Message Editor 1 interprets the final search range vector and generates the display image.

Figure 4.3 illustrates the flow of control among the modules used for the function 1.

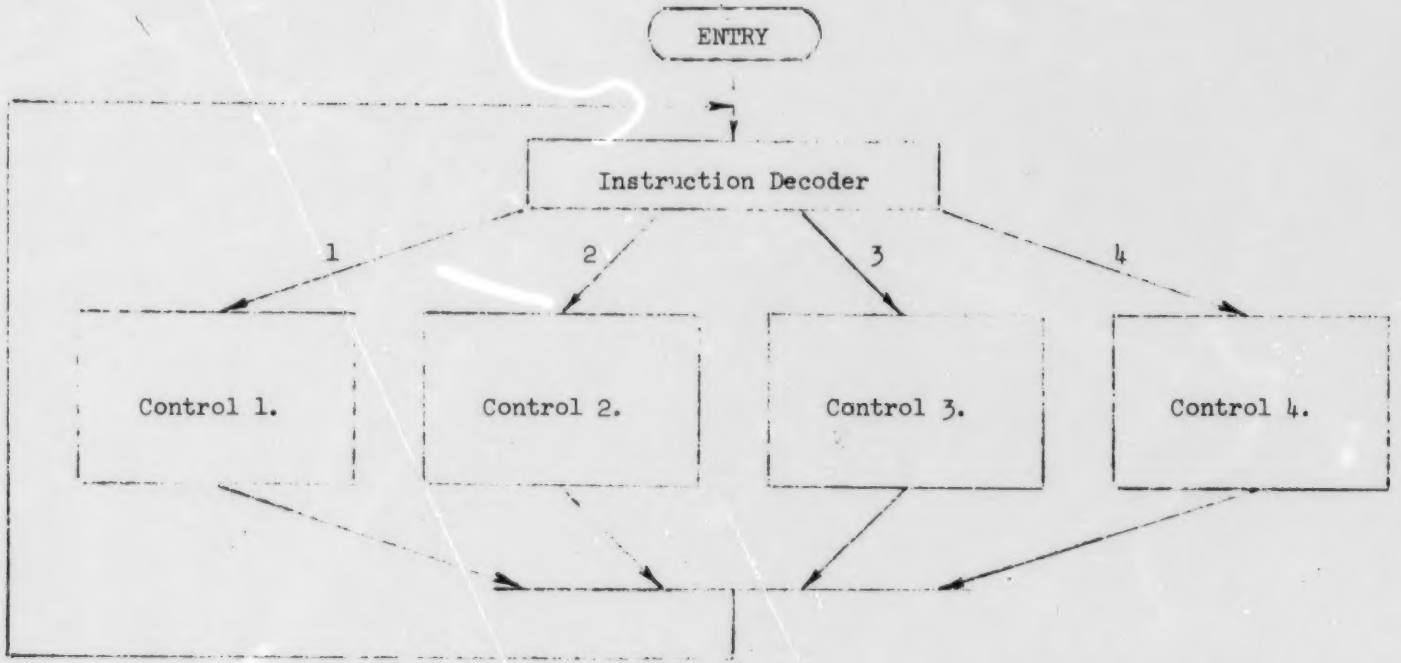


Figure 4.1 Main Flow of GUIDE-0

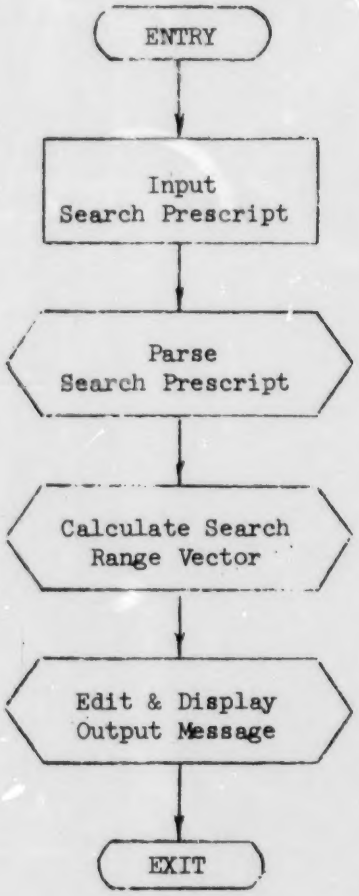


Figure 4.2 Flow of Controller for Function 1.

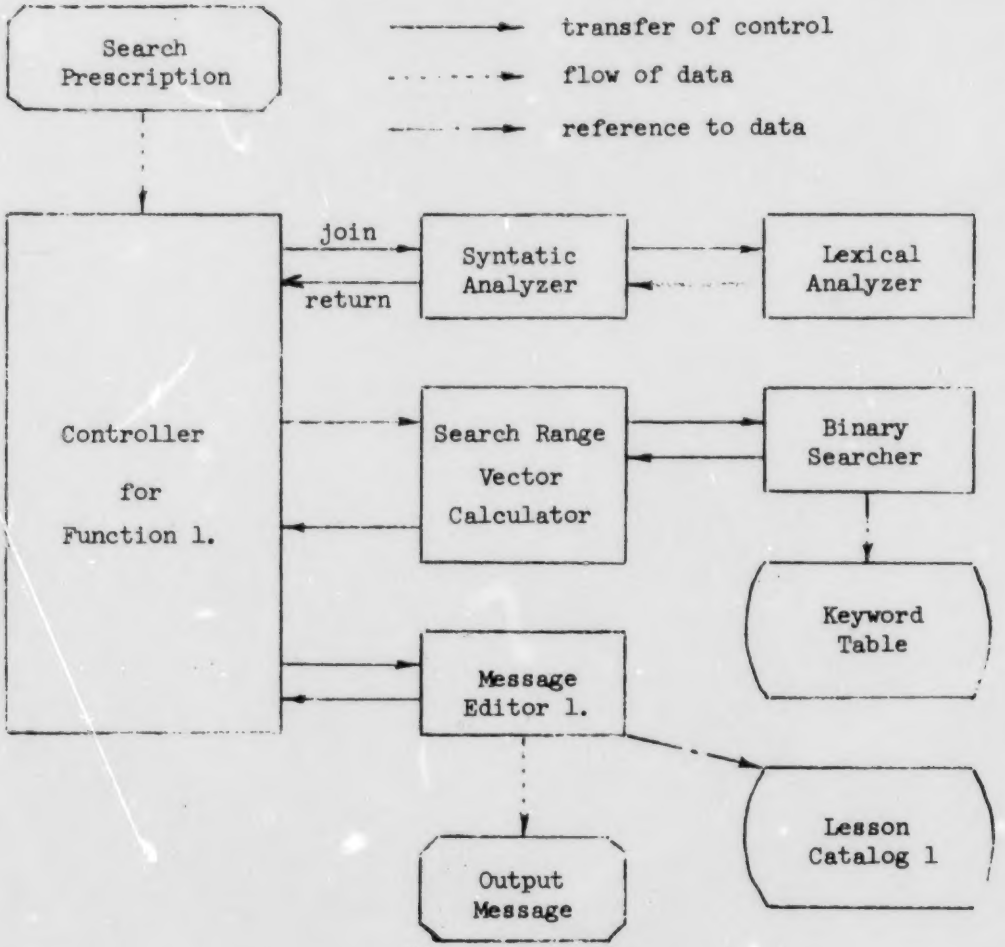


Figure 4.3 Structural Relationship between Modules - 1.

1.3 Controller 2 (Display Lesson Descriptors) "idcdld"

As shown in Figure 4.4 and Figure 4.5, Controller 2 receives a lesson name, and joins the Binary Search module. The Binary Search module searches the Lesson Catalog 2 for the specified lesson, and returns the "logical" location of the lesson in Lesson Catalog 2.

Then Controller 2 joins the Message Editor for Function 2. The message editor reads the necessary data about the lesson in Lesson Catalogs 1 and 2, and generates the display image.

1.4 Controller 3 (Display Course Outline) "idcdco"

As shown in Figure 4.6 and 4.7, Controller 3 receives a course and section number whose course outline the user wants to know, and then joins the Sequential Search module. The Sequential Search module searches the Course Directory for the specified course and section, and returns the "logical" location of the course and section in the Course Directory.

Next, Controller 3 joins Message Editor 3. Message Editor 3 generates that part of the display image which is unique to the course outline, obtains the location of the course outline from the Course Directory, and then joins the Subeditor which is shared with Message Editor 4.* The Subeditor reads the specified location of the Course Outline and Lesson Catalog 1, and generates the rest of the display image.

1.5 Controller 4 (Display Student Record) "idcdsr"

As shown in Figure 4.8 and 4.9, Controller 4 receives the course and section number in which the student is enrolled, and joins the Sequential Search module. The Sequential Search module searches the Course Directory

---

*The Course Outline and Student Record have the same file structure.

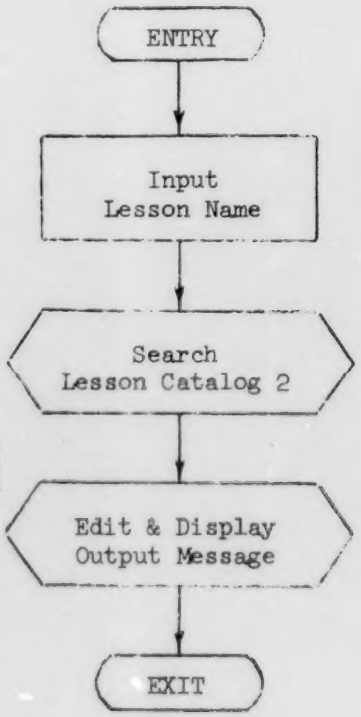


Figure 4.4 Flow of Controller 2

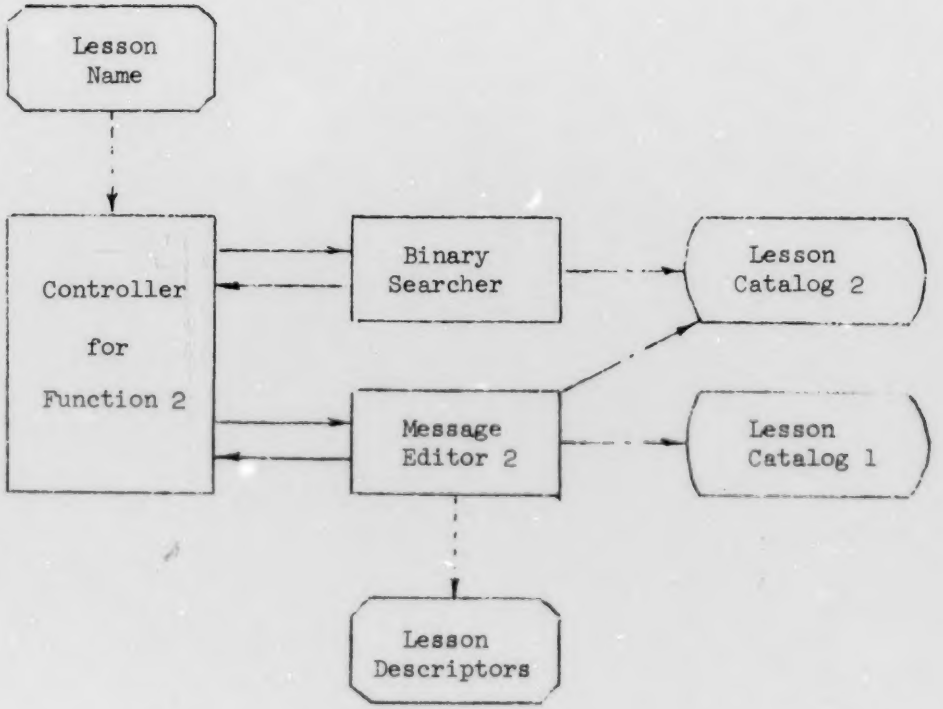


Figure 4.5 Structural Relationship between Modules - 2

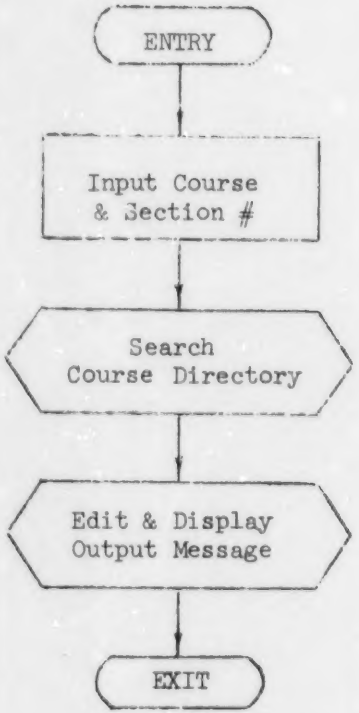


Figure 4.6 Flow of Controller 3

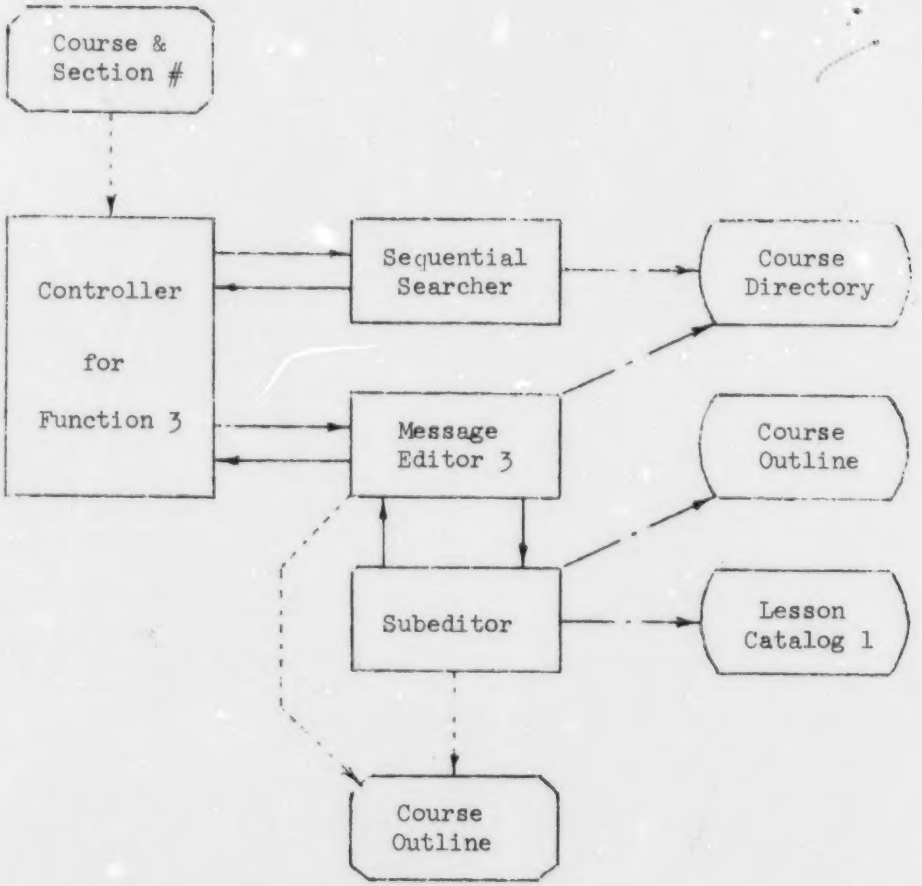


Figure 4.7 Structural Relationship between Modules - 3

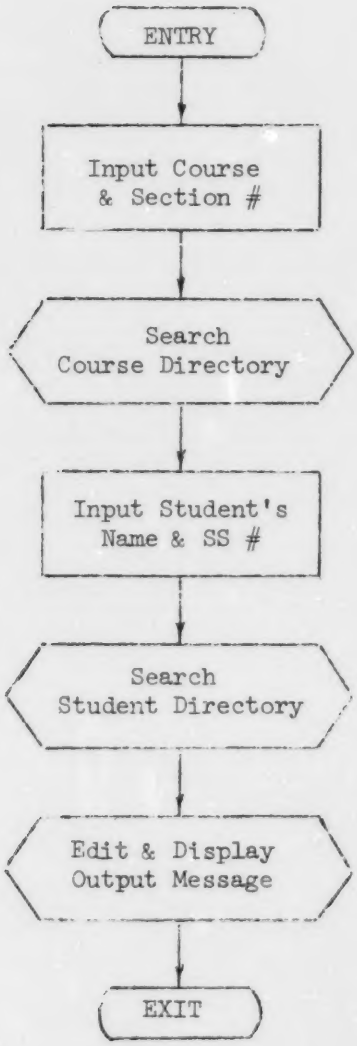


Figure 4.8 Flow of Controller 4

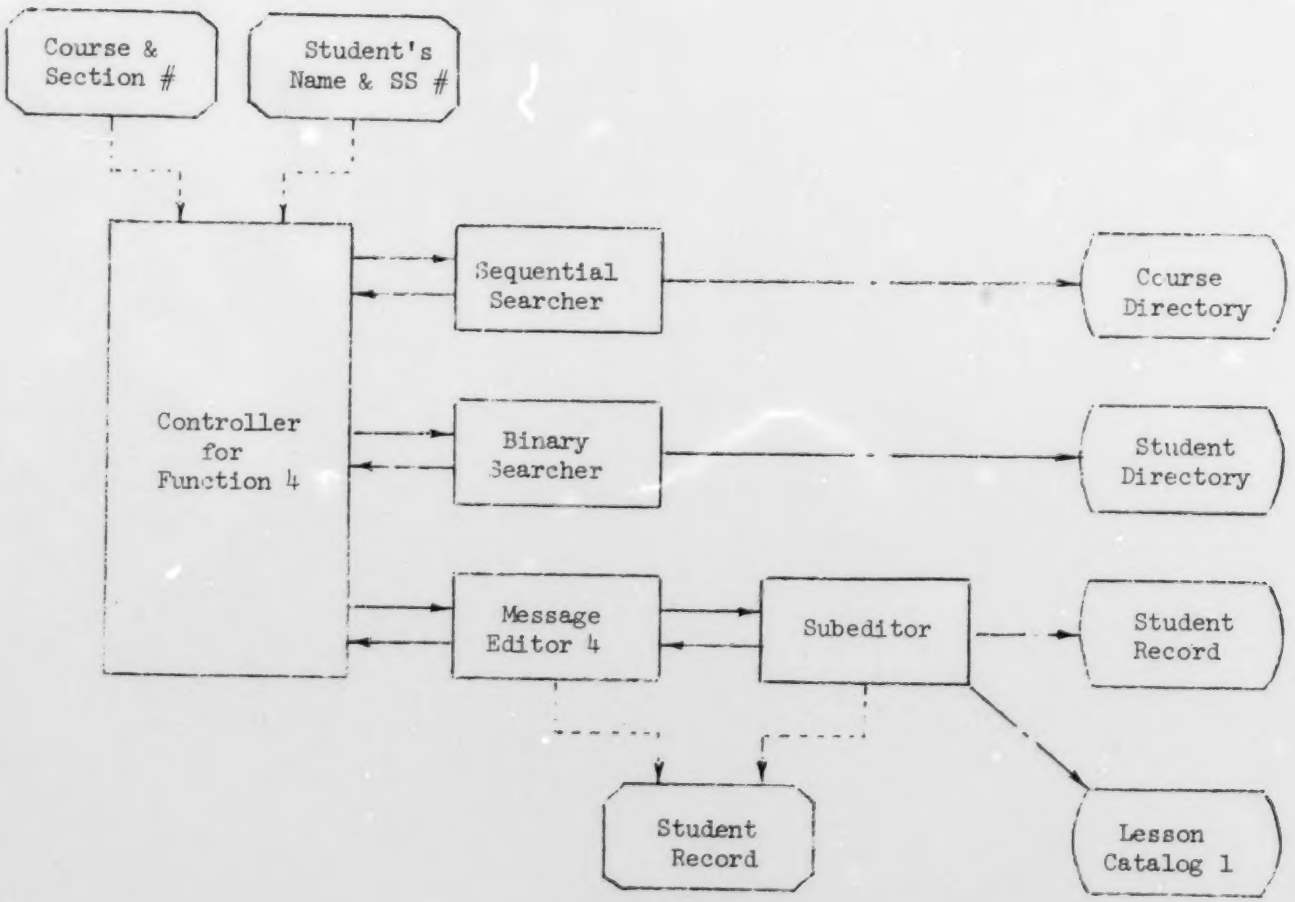


Figure 4.9 Structural Relationship between Modules - 4

for the specified course and section, and returns the "logical" location of the course and section in the Course Directory.

Then Controller 4 receives the student's name and social security number and joins the Binary Search module. The Binary Search module searches the specified (by PSDIRCT and NSTUDNT fields of the Course Directory) portion of the Student Directory for the specified social security number and returns the "logical" location of that number in the Student Directory. Next, Controller 4 joins Message Editor 4. Message Editor 4 generates the part of display image which is unique to the student record, obtains the location and length of the student record in Student Record from the Student Directory and Course Directory (PSRECRD and NLSREC) and joins the Subeditor which is shared with Message Editor 3. The Subeditor reads the specified portion of the Student Record and Lesson Catalog 1, and generates the rest of the display image.

2. Lexical Analyzer for Search Prescription ("lexi")

2.1 General

The Lexical Analyzer is joined by the Syntax Analyzer to get from the search prescription the next token to be analyzed.

2.2 Input

The input is a search prescription (explained in II.1.1) which is stored in the array "spresc(1)" ~ "spresc(lwnpres)". If the search prescription consists of less than 11 characters, then it occupies only the first word of the array, i.e., spresc(1). If it consists of 11 ~ 20 characters, then it occupies the first two words of the array "spresc(1)" and "spresc(2)"; and so on.

2.3 Output

The output is a number stored in "crtoken" which designates one of the operators +, *, ', (,) and ;, if the number is positive, or the "logical" location of the operand (search word) in the search word table "searchw(1)" ~ "searchw(lwschw/2)", if the number is negative (Table 3). The search word table is an array each element of which consists of 2 words (= 20 characters). Thus the maximum length of a search word (a keyword) is 20 characters.

2.4 Functional Description

The Lexical Analyzer ("lexi") examines a search prescription stored in "sprescr(wnspres)" one character at a time. "lexi" has 4 states as shown in Figure 4.10. The operation depends on both the state and the current symbol (character) stored in "tcursym".

(a) State 0 (ready to get a new token)

If the current symbol in "tcursym" is one of the operators (+, *, ', (,) or ;), then "lexi" stores the code of the operator shown in Table 3, into "crtoken" and returns to the Syntax Analyzer.
If the current symbol is a blank, "lexi" ignores it.
If the current symbol is an upper case shift code, "lexi" goes to the State 1. If the current symbol is any other character, the symbol is assumed to be the first character of the new operand (search word or keyword), is stored into the first character position of the new location in the search word table ("searchw(wpsw)"), and goes to State 2.

crtoken	MEANING
< 0	The location of an operand (keyword)
1	The operator + (OR)
2	The operator * (AND)
3	The operator ' (NOT)
4	The operator ;
5	The operator (
6	The operator)

Table 3. Code of Token

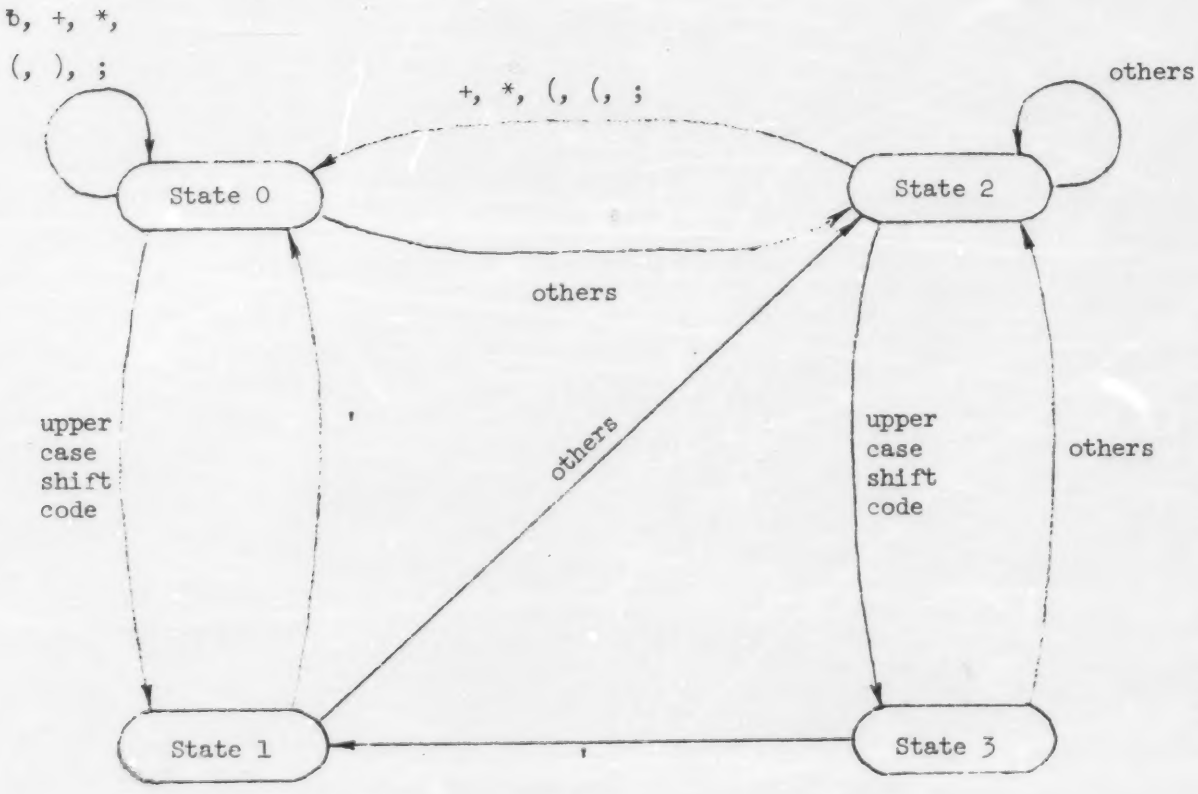


Figure 4.10 State Diagram of Lexical Analyzer

(b) State 1 (upper case character from State 0)

If the current symbol is ', then "lexi" stores 3 into "crtoken", goes to State 0, and returns to Syntax Analyzer.

Otherwise, the symbol is assumed to be the first character of the new operand, is stored into the first character position of the new location in the search word table, and goes to State 2.

(c) State 2 (getting an operand)

If the current symbol is one of the operators, "lexi" backs up one character position in the search prescription so that the current symbol appears again as the next symbol, negates the current location in the search word table and stores it into "crtoken", goes back to State 0, and returns to the Syntax Analyzer.

If the current symbol is the upper case shift code, "lexi" goes to State 3.

Other wise, the current symbol is assumed to be a character of the current operand, and is stored into the current character position of the current location in the search word table.

(d) State 3 (upper case from State 2)

If the current symbol is ', "lexi" backs up one character position in the search prescription so

that ' appears again as the next symbol, negates the current location in the search word table and stores it into "crtoken", goes to State 1, and returns to the Syntax Analyzer.

Otherwise, "lexi" assumes the current symbol is a character of the current operand, stores it into the current character position of the current location in the search word table, and goes back to State 2.

3. Syntax and Semantics Analyzer of Search Prescription ("parser")

3.1 General

The Syntax and Semantics Analyzer ("parser") analyzes a search prescription according to the simple operator precedence grammar [6] described below, and outputs the result in the form of Polish postfix notation.

3.2 The Grammar of the Search Prescription

<Search Presc> ::= <Expression>;

<Expression> ::= <Expression> + <Term> | <Term>

<Term> ::= <Term> * <Factor> | <Factor>

<Factor> ::= <Factor> ' | <Primary>

<Primary> ::= (<Expression>) | <identifier>

3.3 Precedence Relations Between Operators

The meaning of the symbol in Table 4 is as follows:

R > S -- R has precedence over S

R = S -- R and S have the same precedence

R < S -- S has precedence over R

The numbers 2, 3, ..., 9 indicate no relations.

	+	*	'	;	(	)	i
+	>	<	<	>	<	>	<
*	>	>	<	>	<	>	<
'	>	>	>	>	2	>	3
;	<	<	<	=	<	4	<
(	<	<	<	5	<	=	<
)	>	>	>	>	6	>	7
i	>	>	>	>	8	>	9

Table 4. Precedence Matrix

3.4 Input

The input to "parser" is a sequence of tokens extracted from a search prescription by "lexi" as described in the preceding section.

3.5 Output

The output of "parser" is a search prescription transformed into Polish postfix notation. This is stored in the array "postfix(1)" ~ "postfix(lpstfix)". Each element of the array contains either an operator or an operand. Operators are encoded as shown in Table 3. Operands in the postfix notation are the "logical" locations of the operands in the search word table.

3.6 Functional Description

The main frame of the "parser" consists of the following:

(a) Precedence Matrix "precdnc(1)" ~ "precdnc(49)"

(b) The current token "crtoken"

(c) Unreduced tokens in the syntax stack "pstack(1)" ~ "pstack(lpstack)"

(d) The parsed result in "postfix(1)" ~ "postfix(lpstfix)".

"parser" first joins the "lexi" and gets the current token. If the current token in "crtoken" is an operand, the contents in "crtoken" is saved in "crident" and the code designating identifier is assigned into "crtoken". If the current token is an operator, no operation is done in this stage.

Next, the precedence is examined between the top-most operator in the syntax stack "pstack(j)" and the current token in "crtoken". If the operator in the stack has the precedence over the current token, "parser" keeps examining the precedence between the next top-most operator and the current token until it finds the head of the prime phrase to be reduced, and reduces the prime phrase according to the grammar described in 3.2, producing the postfix notation appropriately. If the operator in the stack does not have the precedence over the current token, the contents in "crtoken" is stored at the top of the syntax stack "pstack(i)".

"parser" repeats these operations until it encounters ;.

In the precedence matrix "predmc(1)" ~ "predmc(49)", -1 denotes <, 0 denotes =, and 1 denotes >. Other numbers denote that no relation exists between the two operators.

4. Search Range Vector Calculator "calcsrv"

4.1 Search Range Vector

The Search Range Vector is a bit-string which represents a set of lessons. Each bit of the string corresponds to a lesson in the data base. In the current implementation of GUIDE-0, the Search Range Vector consists of 60 bits representing 60 lessons. For example, the following search range vector

$$\underbrace{01100010 \ldots\ldots 010}_{60\text{-bit}}$$

represents the lessons #2, #3, #7 and #59. Lesson #2 means the lesson whose lesson number, i.e., the "logical" address of the lesson in Lesson Catalog 1, is 2.

4.2 Basic Function of "calcsrv"

The Search Range Vector Calculator "calcsrv" evaluates an expression which represents a search prescription in the form of Polish postfix notation (this expression is generated by the Parser). The result is expressed in the form of a search range vector, and is given to Message Editor 1.

4.3 Input

The input to "calcsrv" is a logical expression which represents a search prescription in the form of Polish postfix notation. The expression is generated by the Parser and is stored in the array "postfix(pointps)". See IV.3.5 for more detail.

4.4 Output

The output of "calcsrv" is a search range vector stored in an element of the array "tsrange(1)" ~ "tsrange(ltsrnge)". The location of the element where the search range vector is stored is given by "pstack(1)".

4.5 Functional Description of "calcsrv"

The Search Range Vector Calculator "calcsrv" is basically a conventional postfix expression interpreter. "calcsrv" scans the "postfix(pointps)". If it encounters an operand, it stores the operand into the stack "pstack(j)". If it encounters an operator, it executes the operation specified by the operator on the rel vant operands which have been stored on the top locations of the stack "pstack", and then stores the result on the top of the stack.

There are two kinds of operands: one is a keyword stored in the search word table "searchw(1)" ~ "searchw(lwpsw)", and the other is a search range vector stored in the search range vector table "tsrange(1)" ~ "tsrange(ltsrnge)". In the "pstack", a keyword is represented by the negated "logical" address of the keyword in the search word table "searchw", while a search range vector is represented by

> the negated "logical" address of the search range vector in the search range vector table "tsrange" minus "lwpsw"

where "lwpsw" is the maximum number of keywords which can be stored in the search word table "searchw(1)" ~ "searchw(lwpsw)". Thus if $-\text{lwpsw} \leq \text{pstack}(j) \leq -1$, then pstack(j) represents a keyword which is stored in "searchw(-pstack(j))", while if $\text{pstack}(j) < -\text{lwpsw}$, then pstack(j) represents a search range vector which is stored in "tsrange(-pstack(j)-lwpsw)".

Thus, before the operation is executed on the operands, the operands have to be checked to see whether they are keywords or search range vectors.

If the current operand is a keyword, i.e.,

$$-\text{lwpsw} \leq \text{pstack}(j) \leq -1,$$

then the operand first has to be transformed to the corresponding search range vector. The search range vector for a keyword is given by the retrieval code attached to the keyword. Thus, if the operand is a keyword, "calcsrv" searches the keyword table for the keyword and obtains the retrieval code attached to it.

After all the operands relevant to the operation are transformed to search range vectors, the operation (AND, OR, NOT) is executed bit by bit on the operands (which are search range vectors). The result of the

operation (which may be the operand of a further operation) is stored into the search range vector table "tsrange(q)". The location "q" of the result (actually -q-lwpsw) is stored on the tcp of the stack "pstack(j)".

Thus the final result of the calculation is put in the search range vector table "tsrange(-pstack(1)-lwpsw)" in the form of a search range vector which represents a set of lessons which match the given search prescription. The location of the final result in the search range vector is given by -pstack(1)-lwpsw.

5. Sequential Search Module "ssearch"

5.1 General

The Sequential Search Module "ssearch" is a general subroutine which searches specified locations of the common storage area in PLATO IV for a specified keyword of the specified length, and returns the "logical" address of the keyword.

5.2 Input (Parameters)

(a) "sdb" -- Start address of the file

The starting address of the file in the "common" storage area which is to be loaded and searched.

(b) "nwload" -- Physical length of the file

The number of physical words to be loaded and searched.

(c) "flength" -- Logical length of the file

The number of items in the file to be loaded and searched.

(d) "scom" -- Start address of common variable

The starting address of the common variables into which the file is loaded.

(e) "key(1)" ~ "key(2)" -- Keyword

The keyword to be searched for (up to 20 characters). The keywords of less than 20 characters are left aligned in "key(1)" and "key(2)".

(f) "kylengt" -- Length of keyword

The number of characters in the keyword.

(g) "subscrp" -- Expression

The expression to calculate the physical address of the specific field which is to be searched within the common variables. For example, suppose that a file, each element (item) of which consists of three physical words, is loaded into the common variables ncl26 ~ ncl85 (therefore, "scom" = 126, "nwload" = 185-(126-1) = 60, "flength" = "nwload"/3 = 60/3 = 20). Furthermore, suppose that the field to be searched is the second word of each item. Then the expression should be

3*(nl2-1)+126+1 = 3*nl2+124

where nl2(=i) contains the logical address of the items of the file. In other words, the expression maps the logical address (1, 2, 3, ..., 20) into the physical address (127, 130, 133, ..., 184).

(h) "count" -- Character count

The number of characters in the above expression.

5.3 Output

The output is the logical address, i, of the searched item in the file. If not found, a value of 0 is returned.

5.4 Functional Description

The specified file (by "sdb" and "nwload") is loaded into the specified common variables (by "scom"). Then the specified field (by "subscrp" and "kylengt") is searched for the keyword stored in "key(1)" and "key(2)" all through the file sequentially. If an item whose field matches the keyword is found, the logical address i of the item is returned. If not found, a value of 0 is returned.

6. Binary Search Module "bsearch"

6.1 General

The Binary Search Module "bsearch" is a general subroutine which searches the specified fields of the specified locations of the common storage area for a specified keyword of the specified length, and returns the "logical" address of the item whose specified field matches the keyword. If an item which matches the keyword cannot be found, a "would-be-address" is returned in negative form. The items are supposed to be sorted lexicographically by the specified field in order that the binary search[7] can be executed.

6.2 Input (Parameters)

In addition to the input to the Sequential Search Module "ssearch", the following input is necessary:

"bsmask" -- Mask pattern for the keyword

The mask pattern for the last "physical" word of the keyword. For example, if the keyword consists of 17 characters, the first 10 characters are contained in KEY(1) and the last 7 characters are contained in KEY(2). Thus, the "bsmask" should contain 00$\underbrace{77777777777777}_{2x7}$0000,

i.e., the 6-bit right shifted mask pattern for 7 characters. If the keyword consists of 6 characters, "bsmask" would be 00$\underbrace{777777777777}_{2x6}$000000.

6.3 Output

The output is the logical address, i, of the searched item in the file whose specified field matches the keyword. If the item matching the keyword is not found, the "would-be-address" of the searched item in the file is returned in the negative form.

6.4 Functional Description

The specified (by "sdb" and "nwload") file is loaded into the specified (by "scom" and "nwload") common variables. The starting address for the binary search is calculated by "flength". Then the specified (by "subscrp", "kylengt" and "bsmask") field of the address is compared with the keyword. The unit of comparison is a maximum of 9 characters. For example, if the keyword consists of 16 characters, the first 9 characters are compared first. If matching occurs, then the next 7 characters are compared. The comparison is numerical in order to tell the next search location (forward or backward). This is the reason why the unit of comparison is not 10 characters (full word) but 9 characters and operands are right-shifted in order to avoid a possible negative value.

If the item whose field matches the keyword is found, the "logical" address i of the item is returned. If not found, the negated "would-be-address" is returned as the value of i.

7. Message Editors

Message Editors consist of 5 separate modules:

(a) Message Editor 1 "edtlsp"

(b) Message Editor 2 "edtdld"

(c) Message Editor 3 "edtdco"

(d) Message Editor 4 "edtdsr"

(e) Subeditor "edtdata"

The first four correspond to the four functions of GUIDE-O while the last is a common subroutine for Message Editor 3 and 4.

7.1 Message Editor 1 "edtlsp"

Message Editor 1 "edtlsp" is the output message editor for the function 1 (display lessons which match search prescription). First it displays the headings on the top of the screen. Then it obtains the final result (search range vector) of the calculation done by "calcsrv", and displays the lesson names and abstracts of the lessons which are represented by the search range vector. Note that each bit of a search range vector corresponds to a lesson and the bit position shows the lesson number, i.e., the "logical" address of the lesson in Lesson Catalog 1. Thus, "edtlsp" scans the search range vector, detecting the bit positions which contain 1's, and displays the lesson name and abstract fields of the corresponding lessons in Lesson Catalog 1. If the number of lessons represented by the vector exceeds the number which can be displayed at one time on the screen, the lessons are paged and the control of turning pages are done by NEXT and BACK key. If the search range vector represents a null set, the editor displays a message which suggests that the user broadens the search range.

An example of the output is shown in Figure 4.11.

7.2 Message Editor 2 "edtdld"

Message Editor 2 "edtdld" is the output message editor for function 2 (display lesson descriptors). As explained in IV.1.3, the Binary Search module obtains the "logical" address of the specified lesson

LESSON	ABSTRACT
racetrack	Simulation Experiment
somaga	Software Management Game to Teach Programming
montecarlo	Area Calculation by Monte Carlo Method

Figure 4.11 Output Format for the Function 1

in Lesson Catalog 2. Then "edtdld" gets the lesson number of the lesson, i.e., the "logical" address of the lesson in Lesson Catalog 1, from the lesson number field "glesnn" in Lesson Catalog 2.

Now since "edtdld" knows both "logical" addresses of the lesson in Lesson Catalog 1 and 2, it displays the necessary fields of the lesson in Lesson Catalog 1 and 2 with the corresponding titles.

An example of the output is shown in Figure 4.12.

7.3 Message Editor 3 "edtdco"

Message Editor 3 "edtdco" is the output message editor for function 3 (display course outline). Since the file structures of the Course Outline and the Student Record are identical and also the output message formats for the function 3 and 4 are essentially the same, Message Editor 3 and Message Editor 4 use the common subroutine Subeditor "edtdata" for displaying the data in the Course Outline and the Student Record.

"edtdco" displays the headings which are unique to function 3. Then it obtains the "logical" pointer and the "logical" length of the course outline of the specified course from the Course Directory. After this, it calculates the starting "physical" address and the "physical" length of the course outline to be loaded. Note that the "logical" address of the course in the Course Directory is given by the Sequential Search module. Then the Subeditor "edtdata" is joined to display the data in the Course Outline.

An example of the output is shown in Figure 4.13.

7.4 Message Editor 4 "edtdsr"

Message Editor 4 "edtdsr" is the output message editor for function 4 (display student record). Like "edtdco", it first displays the

LESSON NAME: plldo Type: exercise

ABSTRACT: Introduction to PL/1 DO-statement

CATEGORY: 2.1 , 3.63

KEYWORDS: iteration flow of control

pl/1

TIME REQUIRED: 40 min.

PREREQUISITES:

plldata	Beginning Computer Science Lessons
pllio	PL/1 Input/Output
pllarray	Introduction to PL/1 Arrays

SEQUELS:

pllif	PL/1 IF-THEN-ELSE Statements

Figure 4.12 Output Format for the Function 2

LESSON	TYPE	PERFORM EXPCTD	TIME REQRD	DATE
racetrack	game		20 min.	9/10/73
plldata	exercise	70	40 min.	9/15/73
pllops	exercise	70	40 min.	9/25/73
pllarray	exercise	65	40 min.	10/10/73
plldo	exercise	70	40 min.	10/20/73
exam	exam	70	50 min.	10/25/73
pllif	exercise	65	60 min.	11/5/73

Figure 4.13 Output Format for the Function 3

headings which are unique to function 4, and then calculates the "physical" address and length of the student record which is to be displyaed. The "physical" address is deduced from the "logical" pointer to the student record, which is obtained from the "psrecrd" field of the Student Directory. The "physical" length comes from the "logical" length of the student record given by Controller 4. Then it joins the Subeditor "edtdata" to display the data in the student record.

An example of the output is shown in Figure 4.14.

7.5 Subeditor "edtdata"

Subeditor "edtdata" is the common subroutine which is used by both the Editor 3 and 4 to display the data stored in the Course Outline or the Student Record. "edtdata" first loads the specified part of the Course Outline or the Student Record and the Lesson Catalog 1. Note that both files have the identical file structure. Then "edtdata" displays the data in every field of the Course Outline or the Student Record except for the lesson number field. The lesson number is used to obtain the lesson name from Lesson Catalog 1.

8. Miscellaneous Modules

The following modules are usually deactivated and should be activated only for very special occasions such as the change of the specification of GUIDE-0 itself or the introduction of the new modules to GUIDE-0 system.

8.1 "inprecd" -- Initialize the Precedence Matrix

This module is used to initialize the precedence matrix "predmc(1)" ~ "predmc(49)" for the "parser". The numbers and their meanings are as follows on page 51.

LESSON	TYPE	PERFORMANCE	TIME SPENT	DATE
racetrack	game		30 min.	9/5/73
plldata	exercise	85	40 min.	9/10/73
pllops	exercise	65	30 min.	9/22/73
pllarray	exercise	80	55 min.	10/7/73
plldo	exercise		min.	/ /
exam	exam		min.	/ /
pllif	exercise		min.	/ /

Figure 4.14 Output Format for the Function 4

-1 -- <

0 -- =

1 -- >

2 ~ 9 -- no relation

8.2 Setting the Experimental Data into the Data Base

The module names and their corresponding file names are listed in Table 5.

MODULE	FILE
debug1	Lesson Catalog 2
debug3	Keyword Table
debug6	Course Directory
debug7	Student Directory
dbgclg1	Lesson Catalog 1
dbgsco	Course Outline
dbgssr	Student Record

Table 5. Data Base Initialization Modules

8.3 Displaying the Variables for Debugging Purpose

(a) "debug2"

Displays the keyword stored in the Search Word Table. Used as a subroutine for "debug4".

(b) "debug4"

Displays the contents of the Postfix and the Search Word Table by using "debug5" and "debug2". Used in Controller 1.

(c) "debugs"

Displays the contents of the Postfix. Used as a subroutine for debug4.

(d) "debug5"

Displays the search range vector in the Search Range Vector Table. Used in Controller 1.

V. FUNCTIONAL SPECIFICATION OF THE DATA BASE EDITOR

GUIDE-O File Editor consists of two parts, Lesson Catalog Editor and Course Record Editor, corresponding to the structure of the files of GUIDE-O explained in the previous chapter.

In the following specification the necessary functions are listed in somewhat random fashion. These functions may be divided into subfunctions or be synthesized into more comprehensive functions to achieve the effective maintenance of the GUIDE-O files.

1. Lesson Catalog Editor

1.1 Addition of a New Lesson Name intc Lesson Catalog

(a) Insert a new lesson name at the first available location in Lesson Catalog 1.

(b) Clear* the corresponding Subject Category and Keyword fields in Lesson Catalog 1.

(c) Insert the new lesson name into Lesson Catalog 2 in such a way that the resultant Lesson Catalog 2 is sorted lexicographically by lesson names.

(d) Set the corresponding "Lesson Number" field in Lesson Catalog 2 to the lesson number of the new lesson (this is the pointer to the new lesson in Lesson Catalog 1 or the "logical" location of the new lesson Lesson Catalog 1,

---

*If the field to be "cleared" is a character string, "clear" means to store "blank" characters (octal 55) into the field. Otherwise "clear" means to set the field to 0.

i.e., the lesson number is an integer between 1 and 60).

(e) Clear the corresponding "Time Required", "Relation to Other Lessons" and "Type" fields.

1.2 Replacement of Abstract

Replace an abstract in the Abstract field of Lesson Catalog 1 of the lesson specified (by lesson name).

1.3 Addition of a Subject Category Code

Add a subject category code into the Subject Category field of the specified lesson, if the field is not full.

1.4 Deletion of a Subject Category Code

Delete a specified subject category code in the Subject Category field of the specified lesson. "Delete" means to change the specified code into blank characters.

1.5 Replacement of a Subject Category Code

Replace the specified subject category code of the specified lesson with the specified subject category code (the combination of 1.3 and 1.4).

1.6 Addition of a Keyword

(a) Test whether or not the space is available for the addition of the specified keyword in the Keyword Field of the specified lesson in Lesson Catalog 1. If not available, no operation.

(b) If the space is available in Keyword field in Lesson Catalog 1, search the Keyword Table for the specified keyword. If the keyword is found, modify the corresponding retrieval code in Retrieval Code field of Keyword Table (set to 1 the bit which corresponds to the specified lesson).

If the keyword is not found and if space is available in the Keyword Table, insert the keyword into the Keyword Table so that the resultant Keyword Table is sorted lexicographically by keywords, set the corresponding retrieval code to all 0's but one bit which corresponds to the specified lesson, and modify all the keyword identification codes in the Keyword field of Lesson Catalog 1 which correspond to the keywords located after the newly inserted keyword in the Keyword Table (add one to all the codes).

If the space is not available, i.e., 255 keywords have already occupied Keyword Table, then no operation.

(c) If the specified keyword is found or inserted in Keyword Table, add the identification code of (the "logical" location of or the "logical" pointer to) the keyword (8-bit code representing 1 ~ 255) into Keyword field of the specified lesson in Lesson Catalog 1.

1.7 Deletion of a Keyword

(a) Search the Keyword Table for the specified keyword and obtain the corresponding identification code.

Note that the identification code is the logical location of the keyword in the Keyword Table and is not actually stored in the Keyword Table.

(b) Modify the corresponding retrieval code (set to 0 the bit which corresponds to the specified lesson). If the resultant retrieval code is all 0's, i.e., if no lessons are related to the keyword, delete the keyword from the Keyword Table, relocate all the keywords which are originally located after the deleted keyword, and modify all the identification codes of the keywords which are relocated in the Keyword field of Lesson Catalog 1.

(c) Delte the identification code of the keyword obtained in (a) from the Keyword field of the specified lesson in Lesson Catalog 1.

1.8 Replacement of a Keyword

Combination of 1.6 and 1.7.

1.9 Addition of a Relation to Other Lesson

Add the specified relation pair consisting of a relation code and a lesson number into the Relations to Other Lessons field of the specified lesson in Lesson Catalog 2, if the field is not full (i.e., the field contains less than - relation pairs).

1.10 Deletion of a Relation to Other Lesson

Delete the specified relation pair from the Relations to Other Lessons field of the specified lesson in Lesson Catalog 2.

1.11 Replacement of a Relation to Other Lesson

Combination of 1.9 and 1.10.

1.12 Replacement of Lesson Type

Replace the 4-bit lesson type code in Type field of the specified lesson in Lesson Catalog 2 with the 4-bit code of the specified lesson type.

1.13 Replacement of the Time Required

Replace the time required in Time Required field of the specified lesson in Lesson Catalog 2 with the specified time required.

2. Course Record Editor

2.1 Registration of a Course

Reserve spaces for a course outline and a student directory of the course.

(a) Search the Course Directory for the specified course and section. If found, then error.

(b) Insert the course and section number into Course and Section Number field at the first available space in the Course Directory.

(c) Reserve the specified amount (the maximum number of lessons) of space in Course Outline, and store the "logical" pointer to and the length of (i.e., the maximum number of lessons which can be contained in the course outline) the space into Pointer to Course Outline and Length of Course Outline fields in the Course Directory.

(d) Store the specified length of the student record (the maximum number of lessons which can be

recorded in the student record) of the course into Length of Student Record field in the Course Directory. Note that the spaces for the student record are not reserved at this stage.

(e) Reserve the specified amount (the maximum number of students to be registered in the course) of space in Student Directory, clear Social Security Number field of the reserved space, and store the "logical" pointer and length of the space into Pointer to Student Directory and Length of Student Directory fields of the Course Directory.

(f) Set to 0 the Number of Students field of the Course Directory.

2.2 Deletion of a Course

Search Course Directory for the specified course and section number. If not found, no operation. If found, change the course and the section number in the Course and Section Number field to a string of blank characters.

2.3 Insertion of a Lesson into Course Outline

Insert a specified lesson (Lesson Number, Date and Performance Required) after the specified line in the course outline of the specified course. Note that the user doesn't specify the Lesson Number but Lesson Name. The editor must find out the lesson number of the specified lesson.

2.4 Deletion of a Lesson from Course Outline

Delete the lesson at the specified line of the course outline of the specified course, and relocate the lessons which are located after the deleted lesson.

2.5 Replacement of a Lesson in Course Outline

Replace the lesson at the specified line of the course outline of the specified lesson with the specified lesson (Lesson Number, Date and Performance).

2.6 Registration of a Student into a Course

(a) Search the student directory of the specified course for the specified social security number. If found, error.

(b) Insert the specified social security number and the student's name at the numerically ordered (by social security number) position in the student directory of the specified course.

(c) Reserve a specified (by Length of Student Record field of the specified course in Course Directory) amount of space in Student Record for the specified student and store the "logical" location of the reserved space into Pointer to Student Record field of the Student Directory.

(d) Copy lesson numbers in the course outline of the specified course into the corresponding field of the student record just reserved above.

If the student record has more space than the corresponding course outline, the rest of the space (Lesson Number field) must be cleared.

2.7 Deletion of a Student from a Course

(a) Search the student directory of the specified course for the specified social security number. If not found, no operation.

(b) Delete all the items (Social Security Number, Student Name, Pointer to Student Record) of the specified student from the student directory of the specified course, and relocate the items which are located after the deleted student.

2.8 Modification of "Date" and "Performance" of Student Record

This function must be activated by the system program of PLATO IV system, i.e., whenever a student termirates a lesson, this function is executed.

(a) Check the lesson type of the lesson the student just finished. If the lesson was an exam type and the lesson had been taken before by the same student, then do nothing. Otherwise,

(b) Modify Date, Performance and Time Spent field of the student record of the lesson in the student record of the student in the specified course.

2.9 Garbage Collection

Garbage collection is activated (called) if the space is not available when Course Record Editor tries to allocate the space for course outline, student directory or student record.

The garbage collection of course outline, student directory and student record requires the following two functions:

(a) Formation of storage map

Form the storage map which tells the current status of storage usage by scanning the Pointer to Course Outline and the Length of Course Outline

fields of Course Directory for the garbage collection of Course Outline (Pointer to Student Directory and Length of Student Directory fields for Student Directory; Pointer to Student Directory, Length of Student Directory and Length of Student Record fields of Course Directory and Pointer to Student Record field of Student Directory for Student Record).

(b) Reallocation of storage

Reallocates all the spaces currently used to the top of the storage and make a big available space at the bottom of the storage.

LIST OF REFERENCES

[1] Nievergelt, J. and Reingold, E. M., "Automating Introductory Computer Science Courses" in the proceedings of the 3rd ACM-SIGCSE Symposium on Computer Science Education (1973), 24-25.

[2] Alpert, D. and Bitzer, D. L., "Advances in computer based education", Science 167 (1970), 1582-1590.

[3] Pradels, J., "The GUIDE", (Report for Ph.D. Preliminary Examination), Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois.

[4] Shirer, D. and Sherwood, B., "aid1", PLATO IV lesson, Computer-based Education Research Laboratory, University of Illinois at Urbana-Champaign, Urbana, Illinois.

[5] Sherwood, B. et al., "aid2", PLATO IV lesson, Computer-based Education Research Laboratory, University of Illinois at Urbana-Champaign, Urbana, Illinois.

[6] Gries, D., "Compiler Construction for Digital Computers", Chapter 6, John Wiley & Sons, Inc., (1971), 122-132.

[7] Knuth, D. E., "The Art of Computer Programming", Volume 3, Chapter 6, Addison-Wesley (1969), 406-422.

APPENDIX

LISTING OF GUIDE-0

```
LESSON GUIDE    AT    08.59.28.    ON    08/24/73

------------------------------------------------------------------------------------------------------------------------ NOTES
 2  *
 3  *  *TO*: *GEORGE      8/24/73
 4  *     UTILITY AND CATALOG ARE MOVED TO LESSON *.GUIDE*. WHICH
 5  *     HAS THE SAME CHANGE CODE AS *.GUIDE0*. PLEASE WORK ON
 6  *     *.GUIDE*. HEREAFTER.  ---  *SHINICHI
------------------------------------------------------------------------------------------------------------------------ DEFINE1
 8  *
                                                                                                        DEFINE1
 9  UNIT    DEFINE1
10  COMMON  GUIDEDATA,GDATA,ALIAS,NO LOAD
11  DEFINE  FIELDNM
12          SAS=177AB, SAR=2254
13          SP1=12, SP2=18, SP3=24, SP4=9, SP5=6, SP6=4
14          MASK1=07777, MASK2=077, MASK3=0777777, MASK4=0777
15          MASK5=077777777777700000000, MASK6=0377, MASK7=017
16          MASK8=07777777777777777777
17          MASK9=077777777777774000000
18          MASK10=077777777777777770000
19          MASKB=0177, MASKC=037, MASKD=037777777777
20          MASKF=077777777777777000000
21  *
22  *  DATA BASE FOR GUIDE0
23  *    LESSON CATALOG1
24          WPIC1=9    $$ NO OF WORDS PER LESSON
25          SDC1=1501  $$ STARTING ADDRESS OF LESSON CATLOG1 IN
26  *         THE COMMON STORAGE
27          SC1=0    $$ STARTING ADDRESS OF LCATALOG1
28          LESSNM1(I)=NC(WPIC1*(I-1)+SC1+1)
29          ABSTRCT(I)=NC(WPIC1*(I-1)+SC1+2)
30          LABSTRC=42    $$ MAX NO OF CHARACTERS OF ABSTRACT
31          SUBJCAT(I)=NC(WPIC1*I,SC1-1) $CLSS 12 $MASKS MASK10
32          NSUBJCT=2   $$ MAX NO OF 5 CATEGORY CODES IN SUBJCAT
33          NDSUBJC=4   $$ MAX NO OF BITS OF SUBJ CATEGORY CODE
34          KEYCODE(I)=NC(WPIC1*I+SC1)
35          NKEYWRD=7   $$ MAX NO OF KEYWORD ID CODE IN KEYCODE
36          NBKEYCD=8   $$ NO OF BITS OF KEYWORD ID CODE
37          NICATLG=A0  $$ MAX NO OF LESSONS IN CATALOG
38          LC1=540     $$ NO OF WORDS OF LCATALOG1
39  *    LESSON CATALOG2
40          SDC2=2041  $$ START ADDRESS OF LESSON CATLOG2 IN THE
41  *         COMMON STORAGE
42          SCATLG2=SC1+LC1
43          NWLCLG2=2   $$ NO OF WORDS PER LESSON OF LCATALOG2
44          LESSNM2(I)=NC(NWLCLG2*I,SCATLG2-1)    $$ LESSON NAME
45          GLESSNN(I)=NC(NWLCLG2*I+SCATLG2) $CLSS 6 $MASKS MASK2
46  *         LESSON NO
47          GTIME(I)=NC(NWLCLG2*I+SCATLG2) $CLSS 12 $MASKS MASK2
48          RELATN(I)=NC(NWLCLG2*I+SCATLG2) $CLSS 12 $MASKS MASK9
49          GTYPE(I)=NC(NWLCLG2*I+SCATLG2) $MASKS MASK6
50          NICTLG2=11    $$ NO OF ITEMS IN LCATALOG2
51  *  THE ABOVE VALUE IS FOR DEBUG(SHOULD BE 10)
```

```
      LESSON GUIDED        AT     08.59.28.    ON     06/24/73

52          LCATLG2=120
53    *     KEYWORD TABLE
54          SDKY=2141   $$ START ADDRESS IN THE COMMON STORAGE
55          SKEYWDT=660. NPRMT=3
56          KEYWRD(I)=NC(3*I+SKEYWDT-2)  $$ KEYWORD TABLE
57          LKEYWRD=20     $$ KEY WORD LENGTH(NO OF CHARACTERS)
58          RETCODE(I)=NC(3*I+SKEYWDT)     $$ RETRIEVAL CODE
59          NIKYWDT=20    $$ NO OF ITEMS IN KEYWORD TABLE
60    *  THE ABOVE VALUE IS FOR DEBUG(SHOULD BE 256)
61          LKEYWDT=768

------------------------------------------------------------------------------------------------------------------------------------------------------ DEFINE2
63    *
64    *     COURSE DIRECTORY
65          SDCD=1   $$ START ADDRESS IN THE COMMON STORAGE
66          SCDIRCT=110. NWPLC=3
67          COURSEN(I)=NC(NWPLC*I+SCDIRCT-2)   $$ COURSE NUMBER
68          LCOURCN=9    $$ NO OF CHARACTERS OF A COURSE NAME
69          SECCODE(I)=NC(NWPLC*I+SCDIRCT-1)
70          PCOUTLN(I)=NC(NWPLC*I+SCDIRCT) $CL$$ SP1 $MASK$MASK1
71          LCOUTLN(I)=NC(NWPLC*I+SCDIRCT) $CL$$ SP2 $MASK$MASK2
72          LSPECRD(I)=NC(NWPLC*I+SCDIRCT) $CL$$ SP3 $MASK$MASK2
73          PSDIRCT(I)=NC(NWPLC*I+SCDIRCT) $AR$$ SP2 $MASK$MASK3
74          LSDIRCT(I)=NC(NWPLC*I+SCDIRCT) $AR$$ SP4 $MASK$MASK4
75          NSTUDNT(I)=NC(NWPLC*I+SCDIRCT) $MASK$ MASK4
76          LCDIRCT=30. NICDIRC=10
77    *
78    *     COURSE OUTLINE
79          SCOUTLN=3001   $$ START ADDRESS IN THE COMMON STORAGE
80          SAD=SCDIRCT+LCDIRCT
81          OLESSNN(I)=NC(I+SAD) $CL$$ SP5 $MASK$ MASK2
82          ODATE(I)   =NC(I+SAD) $CL$$ SP5 $MASK$ MASK5
83          OTIME(I)   =NC(I+SAD) $AR$$ SP1 $MASK$ MASK2
84          OPERFRM(I)=NC(I+SAD) $AR$$ SP6 $MASK$ MASKA
85          OTYPE(I)   =NC(I+SAD) $MASK$ MASK7
86          LCD=150
87    *     STUDENT DIRECTORY
88          SSDIRCT=31   $$ START ADDRESS IN THE COMMON STORAGE
89          NPS=3
90          SOCSECN(I)=NC(NPS*I-2)    $$ SOCIAL SECR NO
91          SNAME(I)   =NC(NPS*I-1)
92          PSPECRD(I)=NC(NPS*I) $MASK$ MASK3
93          NSDIRCT=450
94          OLESSNM(I)=NC(NPIC1*(I-1)+291)
95    *     TEMPORARY USE FOR COURSE OUTLINE DISPLAY
96    *     COURSE OUTLINE AND STUDENT RECORD
97          SSPECRD=481   $$ START ADDRESS OF THE STUDENT RECORD
98    *       IN THE COMMON STORAGE
99          NPIC=1
100         CLESSNN(I)=NC(I+LC1) $CL$$ SP5 $MASK$ MASK2
101         YEAR(I)=NC(I*LC1) $AR$$ 9 $MASK$ MASK8
102         MONTH(I)=NC(I+LC1) $AR$$ 5 $MASK$ MASK7
103         DAY(I)=NC(I+LC1) $MASK$ MASK6
104         TIME(I)=NC(I*LC1) $CL$$ 12 $MASK$ MASK2
105         PERFORM(I)=NC(I+LC1) $AR$$ 1A $MASK$ MASK6
```

```
LESSON GUIDE>          AT     08.59.28.    ON     06/26/73

106  *
107  *
108           SECTLG2=2       $$ SIZE OF EACH ELEMENT OF LCATLG2
109           SACTLG2=A63     $$ STARTING ADDRESS OF LCATLG2

------------------------------------------------------------------------------------------------------------------------ DEFINE3
111  *
112  *
114           SPSPRES=542     $$ STARTING WORD OF SEARCH PRESCRIPT
114           SPRESCR(I)=NC(I+SPSPRES)   $$ SEARCH PRESCRIPTION
115           LWHPRES=20      $$ WORD NO OF SEARCH PRESCRIPTION
116           SPSCHWD=SPSPRES+LWHPRES
117           SEARCHW(I)=NC(2*I+SPSCHWD-1) $$ SEARCH WORD TABLE
118           LWSCHWD=40
119           SPPRECD=SPSCHWD+LWSCHWD
120           PRECONC(I)=NC(I+SPPRECD)  $$ PRECIDENCE REL MATRIX
121           LPRECD=49       $$ THE SIZE OF PRECIDENCE MATRIX
122           SPTFIX=SPPRECD+LPRECD
123           POSTFIX(I)=NC(I+SPTFIX)  $$ PARSED SEARCH PRESCRPT
124           LPSTFIX=20
125           STSRNGE=SPTFIX+LPSTFIX
126           TSRANGE(I)=NC(I+STSRNGE)  $$ SEARCH RANGE TABLE
127           LTSRNGE=20
128           SWFL=STSRNGE+LTSRNGE+1
129           SCRFLCH=SWFL+300
130           CRELATN(I)=NC(I+SREL+500)
131           LREL=1A
132           SLTYPE=(C1+LCATLG2+LKEYWD)+1
133           SPLIT=SCRELCH+LREL
134           LTYPE(I)=NC(I+SLTYPE)  $$ LESSON TYPE DECODER
135           LLTYPE=1A
136  *          ITYPE(0) == ITYPE(LLTYPE-1)
137           PSTACK(I)=N(I+50)   $$ SYNTAX STACK(NS1-N60)
138           LPSTACK=10

------------------------------------------------------------------------------------------------------------------------ DEFINE4
140  *
141           KEY(I)=N(I)
142           KYLENGT=N3      $$ KEY LENGTH(NO OF CHARACTERS)
143           SUMSCRP=N4      $$ EXPRESSION TO COMPUTE ACTUAL ADDRS
144           COUNT=N10       $$ LENGTH OF THE EXPRESSION(NO OF CH)
145           CMPILED=N1A     $$ ADDRESS OF THE COMPILED CODE
146           FLENGTH=N11     $$ SIZE OF THE FILE TO BE SEARCHED
147           HSMASK=N21
148           STARTPS=N41     $$ START POSITION OF BINARY SEARCH
149           K=N13           $$ COMPUTED VALUE OF ACTUAL SUBSCRIPT
150           L=N14           $$ INCREMENT FOR BINARY SEARCH
151           P=N15           $$ WORD NO IN THE FIELD OF FILE
152           LESNAME=N17     $$ LESSON NAME TO BE SEARCHED
153           TRYLENG=N18     $$ REMAINING LENGTH OF KEY
154           CURITEM=N19     $$ CURRENT ITEM TO BE TESTED
155           CURKEY=M20      $$ PARTIAL KEY TO BE COMPARED WITH
156           TEMP=N70, TEMP2=N80, TEMP3=N88
157           CRTOKEN=M23     $$ CURRENT TOKEN
158           CRIDENT=N24     $$ CURRENT IDENTIFIER
```

```
        LESSON GUIDED          AT     08.59.28.     ON     06/24/73

213  *
214  *  INSTRUCTION DECODE

215  UNIT     IDECODE                                                    IDECODE
216  AT       505
217  WRITE    *WHICH OF THE FOLLOWING SERVICES DO YOU WANT*/
218  AT       708
219  WRITE    1) SEARCH FOR AND DISPLAY LESSONS WHICH MATCH THE
220             SEARCH PRESCRIPTION
221           2) DISPLAY LESSON DESCRIPTORS
222           3) DISPLAY COURSE OUTLINE
223           4) DISPLAY STUDENT RECORD
224  AT       2005
225  WRITE    *TYPE-IN THE NUMBER.
226  ARROW    2205
227  LONG     1
228  STORE    INSTRCT
229  OR
230  JUMP     INSTRCT-2, IDCLSP, IDCOLD, IDCORD, IDCDSR
231  STUB
232  GOTO     INSTRCT=*,ASH          *, X, ERR*/*/*/*/*/
233  *****
234  START
235  *
236  *
237  *  CONTROLLER 1

238  UNIT     IDCLSP                                                     IDCLSP
239  HELD     HSPRESC
240  DATA     DKEYTBL
241  AT       505
242  WRITE    *TYPE-IN A SEARCH PRESCRIPTION (A LOGICAL EXPRESSION
243           OF KEYWORDS AND/OR PHRASES FOLLOWED BY A SEMICOLON).
244  AT       2505
245  WRITE    *HOF*LOP AVAILABLE.
246           *D*A*T*A TO SEE THE AVAILABLE KEYWORDS.
247  ARROW    805
248  STORE A  SPRESCR(1),LSPRSCR
249  OR
250  UNLOADC SPRESCR(1),SPSPRFS,3001, LWNPRFS
251  JOIN     PARSER
252  *  PARSE THE GIVEN SEARCH PRESCRIPTION
253  *
254  BREAK
255  STUB
256  JOIN     DEBUGA
257  *  DISPLAY POSTFIX(PARSED RESULT)
258  PAUSE
259  AT       2205
260  WRITE    INTERMEDIATE RESULTS
261  START
262  JOIN     CALCSRV
263  *  CALCULATE SEARCH RANGE VECTOR
```

BEST COPY AVAILABLE

```
        LESSON WRITTEN          AT      08.59.28.     ON      04/24/73

264   *
265   STOP
266   JOIN    DEBUGS
267   *  DISPLAY THE FINAL SEARCH RANGE VECTOR
268   PAUSE
269   STAR"
270   BREAK
271   ERASE
272   JOIN    EDTLSP
273   *  EDIT AND DISPLAY THE OUTPUT MESSAGE
274   PAUSE
275   JUMP    IDECODE
276   *
277   *  EXPLANATION OF THE SEARCH PRESCRIPTION(*H*E*L*P UNIT)

278   UNIT    HSPRESC                                                  HSPRESC
279   DATA    OKEYTBL
280   AT      307
281   WRITE   *THE SEARCH PRESCRIPTION IS A LOGICAL EXPRESSION OF
282           KEY*ORDS AND/OR PHRASES FOLLOWED BY A SEMICOLON.
283           *THE ALLOWED LOGICAL OPERATIONS ARE *A*N*D(*), *O*R(*) AND
284           *N*O*T(*7).  *OPERATORS AND OPERANDS MAY BE SEPERATED BY
285           BLANKS, AND PARENTHETICAL EXPRESSION NESTED TO ANY
286           LEVEL IS ALLOWED.
287           *FOR EXAMPLE* SUPPOSE YOU WANT TO STUDY ABOUT DATA
288           STRUCTURES OTHER THAN ARRAYS. IN ANY LANGUAGES OTHER
289           THAN *P*L/1 AND *A*L*G*O*L. *THEN THE SEARCH PRESCRIPTION
290           COULD BE AS FOLLOWS*!
291
292               DATA STRUCTURE*ARRAY*7*/PL/1*ALGOL)*7!
293
294           *BE SURE NOT TO FORGET A SEMICOLON.
295   AT      2507
296   WRITE   *D*A*T*A TO SEE THE AVAILABLE KEYWORDS.
297   END
298   *
299   *  LISTING OF AVAILABLE KEYWORDS

300   UNIT    OKEYTBL                                                  OKEYTBL
301   LVARC   KEYWORD(I), SDKY, LKEYWDT
302   CALC    I = 0
303           K = 105

304   ENTRY   NXTKEYW                                                  NXTKEYW
305   CALC    K = K+100
306   GOTO    K>3000, X, IPLUS1
307   PAUSE
308   ERASE
309   CALC    K = 205

310   ENTRY   IPLUS1                                                   IPLUS1
311   CALC    I = I+1
312   GOTO    I>NUMKYWDT, OKEYTBL, X
```

```
        LESSON GUIDE          AT      08.59.28.     ON      04/24/73

313  AT      *
314  SHOW    KEYWORD(I), 20
315  GOTO    NXTKEYW

                                                                                DKEXIT
316  ENTRY   DKEXIT
317  END
 ---------------------------------------------------------------------------------------------------------- CONTROL
319  *
320  *  CONTROLLER 2

                                                                                IDCOLD
321  UNIT    IDCOLD
322  AT      905
323  WRITE   *WHAT LESSON DO YOU WANT*/
324  ARROW   1105
325  STORE   KEY(1),10
326  OK
327  *  SET PARAMETERS FOR RSEARCH(SEE *.RSEARCH* FOR THE MEANING
328  *  OF EACH PARAMETER)
329  CALC    KYLENGT = 10
330          HSMASK = 07777777777777777777
331          FLENGTH = NTCTLG2
332          COMPILED = 0
333          STARTPS = 1
334          SCOM=SCATLG2+1
335          SDR = SDC2
336          NWLOAD = LCATLG2
337  PACK    COUNT* SUBSCRP. 2*N12+599      ** I=N12
338  JOIN    RSEARCH
339  *  SEARCH THE LESSON CATALOG 2 FOR THE SPECIFIED LESSON
340  ERASE
341  JOIN    EDTOLD
342  *  EDIT AND DISPLAY THE OUTPUT MESSAGE
343  JUMP    IDECODE
344  *
345  *
346  *  CONTROLLER 3

                                                                                IDCOCO
347  UNIT    IDCOCO
348  AT      905
349  WRITE   *TYPE-IN THE COURSE AND SECTION NUMBER(EG. CS105A1).
350  ARROW   1105
351  STORE   KEY(1),ICOURCN
352  OK
353  JOIN    SPCOTRC
354  *  SET PARAMETERS FOR SSEARCH
355  JOIN    SSEARCH
356  *  SEARCH THE COURSE DIRECTORY FOR THE SPECIFIED COURSE AND
357  *  SECTION
358  STOP
359  AT      1505
360  WRITE   I =
361  SHOW    I
```

```
362  *  SHOW THE SEARCH RESULT. IF NEGATIVE, NOT FOUND.
363  PAUSE
364  START
365  BREAK
366  ERASE
367  JOIN     EDTDC0
368  *  EDIT AND DISPLAY THE OUTPUT MESSAGE
369  PAUSE
370  JUMP     IDFCOUF
371  *
372  *  CONTROLLER 4

                                                                                IDCDSR
373  UNIT     IDCDSR
374  AT       505
375  WRITE    *TYPE-IN THE COURSE AND SECTION NUMBER YOU ARE
376           ENROLLED IN(EG. CS101E1).
377  ARROW    705
378  STOREA   KEY(1),LCOURCN
379  OK
380  JOIN     SPCDIRC
381  *  SET THE PARAMETERS FOR SSEARCH
382  JOIN     SSEARCH
383  *  SEARCH THE COURSE DIRECTORY FOR THE SPECIFIED COURSE AND
384  *  SECTION
385  GOTO     I<0,ERNOCRS,X
386  STOP
387  *  DEBUG
388  AT       1505
389  WRITE    I =
390  SHOW     I
391  *  SHOW THE SEARCH RESULT
392  PAUSE
393  START
394  BREAK
395  AT       1505
396  ERASE    1*30
397  LOADC    COURCEN(1),SUCD,LCDIRCT
398  *  LOAD THE COURSE DIRECTORY
399  *  SET THE PARAMETERS FOR THE RSEARCH
400  CALC     STARTPS = PSDIRCT(I)
401           FLENGTH = NSTUDNT(I)
402           SDA = SSDIRCT+(PSDIRCT(I)-1)*WPS
403           NWLOAD = FLENGTH*WPS
404           NLSREC = LSRECRD(I)
405           CMPILED = 0
406           SCOM = 1
407  PACK     COUNT,SUBSCRP,3*N12=2
408  AT       1305
409  WRITE    *TYPE-IN YOUR NAME,(EG. FONDA,JANE)
410  ARROW    1505
411  STOREA   STDNAME,LSNAME
412  OK
413  AT       1705
414  WRITE    *TYPE-IN SOCIAL SECURITY NUMBER(EG. 344520078),
```

```
        LESSON GUIDED          AT      08.59.28.     ON      04/24/73

415  ARROW   JUMP
416  STUDFA  KEY(1),0
417  OK
418  CALC    KYLENGT = 3
419          HSPACK = MASKB
420  JOIN    HSEARCH
421  *  SEARCH THE STUDENT DIRECTORY FOR THE SPECIFIED SOCIAL
422  *  SECURITY NO.
423  *
424  STUD
425  AT      2105
426  WRITE   I =
427  SHOW    I
428  *  SHOW THE SEARCH RESULT
429  PAUSE
430  START
431  BREAK
432  ERASE
433  JOIN    EDITOR
434  *  EDIT AND DISPLAY THE OUTPUT MESSAGE
435  PAUSE
436  JUMP    IDECODE
437  *

438  ENTRY   ERNOCRS                                                    ERNOCRS
439  AT      1005
440  WRITE   *THE COURSE IS NOT INCLUDED IN OUR DATA BASE.
441  PAUSE
442  JUMP    IDECODE
443  *

------------------------------------------------------------------------------------------------------------------------ SLPRFSC
445  *
446  *  SEARCH LESSONS BY KEYWORDS

447  UNIT    CALCSRV                                                    CALCSRV
448  CALC    POINTPS = 1 = 2 = 0

449  ENTRY   INCRP                                                      INCRP
450  LOADC   POSTFIX(), SPTFIX+300, LPSTFIX+LTSRNGE
451  CALC    POINTPS = POINTPS + 1
452  GOTO    POSTFIX(POINTPS),X,EILGINF,BOP,BOP,OPRND2,OPRND2
453  CALC    J = J+1
454  GOTO    J>LPSTACK, EPSTOVF, X
455  CALC    PSTACK(J) = POSTFIX(POINTPS)
456  GOTO    INCRP

457  ENTRY   BOP                                                        BOP
458  GOTO    PSTACK(J-1)+LWRSV < 0, ISRNGE1, X
459  *       SEARCH KEYWORD
460  CALC    TEMP = J-1
461  BREAK
462  JOIN    SETPARM
463  JOIN    HSEARCH
```

```
464  GOTO    I>0. X. NOKEY
465  LOADC   KEYWORD(I). SDKY. LKEYWDT
466  CALC    SRANGE1 = RETCODE(I)
467  GOTO    OPRND2
468  *       OPERAND IS A SEARCH RANGE VECTOR
469  ENTRY   ISRNGE1                                          ISRNGF1
470  CALC    SRANGE1 = TSRANGE(-PSTACK(J-1)-LWPSW)
471  ENTRY   OPRND2                                           OPRND2
472  GOTO    PSTACK(J)+LWPSW<0. ISRNGE2. X
473  *       SEARCH KEYWORD
474  CALC    TEMP = J
475  BREAK
476  JOIN    SETPARM
477  JOIN    HSEARCH
478  GOTO    I>0. X. NOKEY
479  LOADC   KEYWORD(I). SDKY. LKEYWDT
480  CALC    SRANGE2 = RETCODE(I)
481  GOTO    INCRO                                            NOKEY
482  ENTRY   NOKEY
483  DATA    DKEYTBL
484  AT      2501
485  ERASE   120
486  AT      2505
487  WRITE   *AT LEAST ONE OF THE KEYWORDS SPECIFIED IS NOT
488          INCLUDED IN THE *KEYWORD *TABLE.
489  PAUSE
490  JUMP    IDCLSP
491  ENTRY   ISRNGE2                                          ISRNGF2
492  LOADC   TSRANGE(1). STSRNGE+3001. LTSRNGE
493  CALC    SRANGE2 = TSRANGE(-PSTACK(J)-LWPSW)
494  ENTRY   INCRO                                            INCRO
495  LOADC   POSTFIX(1). SPTFIX+3001. LPSTFIX+LTSRNGE
496  CALC    Q = Q+1
497  GOTO    Q>LTSRNGE. ETSPOVF. X
498  GOTO    POSTFIX(POINTPS)-2. X. AND. NEG. SCOLN
499  *  *,OR*.
500  CALC    TSRANGE(Q) = SRANGE1 $UNION$ SRANGE2
501  GOTO    DECRJ                                            AND
502  ENTRY   AND
503  CALC    TSRANGE(Q) = SRANGE1 $MASK$ SRANGE2              DECRJ
504  ENTRY   DECRJ
505  CALC    J = J-1
506  GOTO    STACKO                                           NEG
507  ENTRY   NEG
508  CALC    TSRANGE(Q) = -SRANGE2  ** COMPLEMENT             STACKO
509  ENTRY   STACK1
```

```
510  CALC    PSTACK(1) = -8-LWPSW
511  STOP
512  SHOW    PSTACK(J)
513  SHOWN   TSRANGE(-PSTACK(1)-LWPSW)
514  START
515  IMMLOADC TSRANGE(9), W.GTSRNGE+7000, 1
516  GOTO    INCRP

517  ENTRY   SCOLN                                                    SCOLN
518  GOTO    JP1. SKEXIT. X
519  AT      25.5
520  WRITE   *ERROR*. *ILLEGAL POSTFIX FORMAT.
521  GOTO    SKEXIT

522  ENTRY   EPSTOVF                                                  EPSTOVF
523  AT      25.5
524  WRITE   *ERROR*. PSTACK OVERFLOW DURING SEARCH PROCESS.
525  GOTO    SKEXIT

526  ENTRY   ETSROVF                                                  ETSROVF
527  AT      25.5
528  WRITE   *ERROR*. TSRANGE OVERFLOW.

529  ENTRY   SKEXIT                                                   SKEXIT
530  PAUSE
531  CALC    TSRANGE(0) = SRANGE2
532          PSTACK(J) = -8-LWPSW
533  IMMLOADC TSRANGE(1), STSRNGE+3001, LTSRNGE
534  EXIT    1

535  ENTRY   EILGINF                                                  EILGINF
536  AT      25.5
537  WRITE   *ERROR*. *ILLEGAL INFORMATION IN POSTFIX. *IGNORED.
538  GOTO    INCRP

539  ENTRY   LSRANGE                                                  LSRANGE
-----*----*----*
```

```
        LESSON OUTDEF        AT     08.59.28.    ON     04/24/73


                                                                                                                    SPCDIRC
558  UNIT    SPCDIRC
557  PACK    COUNT,SUBSCRP,TEMP,P,104
558  CALC    FLENGTH = NICDIRC
559          SDR = SDCD
560          KYLENGT = ICOURCN
561          FILE = 4
562          CMPILED = 0
563  EXIT    1

 ------*-*-------------------------------------------------------------------------------------------------------------------------------------------- SEARCH
565  *
566  *  --BINARY SEARCH SUBROUTINE--
567  *
568  *  SET THE FOLLOWING PARAMETERS BEFORE JOINING BSEARCH.
569  *  THE EXAMLE OF THE USAGE OF THE ROUTINE CAN BE FOUND IN
570  *  THE UNIT =INFOLD=(IN BLOCK =INCODE=)
571  *  PARAMETERS:
572  *    KEY(1)  --- THE ITEM TO BE SEARCHED
573  *    KYLENGT --- NO OF CHARS OF KEY
574  *    BSMASK  --- MASK PATTERN FOR THE LAST WORD OF THE KEY
575  *                ER. IF THE KEY CONSISTS OF 17 CHARACTERS
576  *                INCLUDING BLANKS, BSMASK SHOULD CONTAIN
577  *                0007777777777777770000(NOTE THAT RIGHT SHIFT
578  *                ED 6-BIT, NOT 077777777777777000003.)
579  *    SDR  ------ START ADDRESS OF THE FILE TO BE SEARCHED
580  *                IN THE COMMON STORAGE(1 -- 4184)
581  *    SCOM  ----- START ADDRESS OF THE FILE IN THE COMMON
582  *                VARABLES(NC1 -- NC1500)
583  *    FLENGTH --- THE LENGTH OF THE FILE TO BE SEARCHED
584  *    NWLOAD  --- NO OF PHYSICAL WORDS IN THE FILE
585  *    SUBSCRP --- THE STRING FOR CALCULATION OF THE ACTUAL
586  *                SUBSCRIPT OF THE FILE
587  *    COUNT   --- NO OF CHARS OF SUBSCRP STRING

                                                                                                                    BSEARCH
588  UNIT    BSEARCH
589  LOADC   NC(SCOM),SDR,NWLOAD

                                                                                                                    CSITEM
590  ENTRY   CSITEM
591  *  CALCULATE STARTING ITEM NO
592  CALC    TEMP = FLENGTH
593          L = 1
594  30      BRANCH TEMP>1, 10, 20
595  10      L = 2L
596          TEMP = TEMP/2
597          BRANCH 30
598  20      I = L = L/2

                                                                                                                    CMPT
599  ENTRY   CMPT
600  COMPILE K, SUBSCRP, COUNT, CMPILED
601  *    SUBSCRP --- STRING TO BE COMPILED(1ST WORD)
602  *    COUNT ----- NO OF CHARS OR THE STRING
603  *    CMPILED --- POINTER TO COMPILED CODE
```

```
404  CALC    P = -
405          TKYLENG = KYLENGT
406  85      P = P+1
407          TKYLENG = TKYLENG-8
408          BRANCH TKYLENG>F, 10, 20
409  10      CURITEM = NC(K,P-1) $ARSS 6
410          CURITEM = CURITEM $MASK$ 077777777777777777
411          CURKEY = KEY(P) $ARSS 6 $MASK$ 077777777777777777
412          BRANCH 30
413  20      CURITEM = NC(K,P-1) $ARSS 6 $MASK$ RSMASK
414          CURKEY = KEY(P) $ARSS 6 $MASK$ RSMASK
415  30      BRANCH CURITEM-CURKEY, 40, 50, 60
416  50      BRANCH TKYLENG>0, 70, 85
417  70      TKYLENG = TKYLENG-1
418          CURITEM = NC(K,P-1) $MASK$ 07
419          CURKEY = KEY(P) $MASK$ 07
420          BRANCH CURITEM-CURKEY, 40, 90, 60
421  90      BRANCH TKYLENG>0, 85, 85
422  40      BRANCH I>1, 45, 73
423  45      L = L/2
424          I = I+L
425          BRANCH I>FLENGTH, 60, 75
426  60      BRANCH L>1, X, 80
427  65      L = L/2
428          I = I-L
429          BRANCH I>FLENGTH, 60, 75
430  73      BRANCH I=FLENGTH+1, 78, X, 75
431          I = FLENGTH
432  75      TEMP = -1
433          BRANCH 8
434  78      I = I+1
435  80      I = -I
436  85      TEMP = 0
437          BRANCH 8
438  GOTO    TEMP, CHPT, X
439  EXIT    1

440  ENTRY   ENDFILE                                                    ENDFILE
441  AT      3005
442  WRITE   *BINARY SEARCH CANNOT BE DONE ON THE SPECIFIED FILE.
443  CALC    I = 0
444  EXIT    1

------**----------**----------**----------**----------**----------**----------**----------**---------- PARSER
446  *
447  *  SEQUENCIAL SEARCH SUBROUTINE
448  *

                                                                        $SEARCH
449  UNIT    $SEARCH
450  GOTO    FILE66,X
451  LOADC   COURCEN(1),$CDIRCT+1501,LCDIRCT
452  LOADC   COURCEN(1),$OR,LCDIRCT
453  *  LOAD COURCE DIRECTORY
454  CALC    I = 0
```

```
                                                                        NEXTITM
455  ENTRY   NEXTITM
456  CALC    I = I+1
457  GOTO    I>FLENGTH,NFOUND,X
458  COMPUTE X,SUBSCRP,COUNT,COMPILED
459  *  COMPUTE PHYSICAL SUBSCRIPT
460  SEARCH  KEY(I),KYLENGT,NC(K),1,TEMP
461  GOTO    TEMP,NEXTITM,X
462  EXIT    1
                                                                        NFOUND
463  ENTRY   NFOUND
464  CALC    I=0
465  EXIT    1
466  *
467  *
468  *  SYNTAX AND SEMANTICS ANALYSER FOR SEARCH PRESCRIPTION
469  *  UNDER OPERATOR PRECIDENCE GRAMMAR
470  *
                                                                                PARSER
471  UNIT    PARSER
472  DO      CLSCHWD, I=1:20
473  CALC    PSTACK(1) = DELIMIT
474          I = 1
475          STATE = POINTPS = 0
476          WHSPRES = 0  $$ INIT WD NO OF SEARCH PRESCRPT
477          CPSP = 10
478          CPSW = WPSW = 1  $$ INIT CHAR, WORD POS OF SEARCH WD
                                                                        NEWTOKEN
479  ENTRY   NEWTOKEN
480  JOIN    LEXI
481  LOADC   PRECONC(1),SPPRECD+3001,A9
482  CALC    BRANCH CRTOKEN<0, X, 10
483          CRIDENT = CRTOKEN
484          CRTOKEN = CRIDENT
485  10      BRANCH PSTACK(I)>0, X, 20
486          J = I
487          BRANCH 30
488  20      J = I-1
489  30      TEMP = (PSTACK(J)-1)*NOFPROW + CRTOKEN
490          BRANCH PRECONC(TEMP), X, X, 40, 91, 92
491          I = I+1
492          PSTACK(I) = CRTOKEN
493          FCPARSE = -1
494          BRANCH 0
495  40      X = PSTACK(J)
496          J = I-1
497          BRANCH PSTACK(J)>0, 50, X
498          J = I-1
499  50      TEMP = (PSTACK(J)-1)*NOFPROW + 0
500  *       CALCULATE THE ARGUMENT OF THE PRECIDENCE RELATION
501  *       MATRIX
502          BRANCH PRECONC(TEMP), X, 40, 90
503  *       THE HEAD OF THE PRIME PHRASE FOUND
504          BRANCH I-J-2, X, 60, 70
```

```
LESSON BUILDER          AT      08.59.28.     ON      04/26/73

705  *          REDUCE  *F *1*1= 1
706             POINTPS = POINTPS + 1
707             POSTFIX(POINTPS) = CRIDENT
708             HRANCH A0
709  *          REDUCE  *P *1*1= *P*7
710  A0         POINTPS = POINTPS + 1
711             POSTFIX(POINTPS) = PSTACK(I)
712             HRANCH A0
713  70         HRANCH PSTACK(I)=APARENT, A0, X
714  *          REDUCE BINARY OPERATION
715             POINTPS = POINTPS + 1
716             POSTFIX(POINTPS) = PSTACK(I-1)
717  A0         I = I-1
718             PSTACK(I) = NONTERM
719             HRANCH I=2, X, 30
720             BRANCH CRTOKEN=SCOL, X, 30
721             POINTPS = POINTPS + 1
722             POSTFIX(POINTPS) = SCOL
723             ECPARSE = 0
724             BRANCH 0
725  90         ECPARSE = 11
726             HRANCH 0
727  91         ECPARSE = 1
728             BRANCH 0
729  92         ECPARSE = 2
730  UNLOADC    POSTFIX(1),SPTFIX+3001,LPSTFIX
731  GOTO       ECPARSE, NEWTOKEN, X
732  EXIT       1
733  *


734  UNIT       CLSCHWD                                                    CLSCHWD
735  CALC       NC(SPSCHWD+2*I-1) = *,          *,
736             NC(SPSCHWD+2*I) = *,          *,
737  UNLOADC    SEARCHW(I), SPSCHWD+2*I+2000, 2


738  UNIT       DUMMY1                                                     DUMMY1

-------------------------------------------------------------------------------- LEXI
740  *
741  *  LEXICAL ANALYZER FOR SEARCH PRESCRIPTION
742  *     STATE0 --- EXPECTING A NEW TOKEN
743  *     STATE1 --- UPPER CASE SHIFT CODE RECEIVED AS 1ST CODE
744  *     STATE2 --- RECEIVING LETTERS IN A SEARCH WORD
745  *     STATE3 --- UPPER CASE CODE IN ST2


746  UNIT       LEXI                                                       LEXI

747  ENTRY      LPINWRD                                                LPINWRD
748  AREA
749  LOADC      SPRESCH(1), SPSPRES+3001, 1P
750  CALC       CPSP = CPSP+1
751  GOTO       CPSP>10, X, GETNSYM
```

```
752   CALC    #WSPRES = #WSPRFS + 1
753   GOTO    #WSPRES>LWNPRES, LONGS9, X
754   CALC    CWSPRES = $PRESCR(#WSPRFS)
755   CALC    CPSP = 1

756   ENTRY   GETNSYM  $$ GET NEXT LETTER OF SEARCH PRESCRPT
757   CALC    CWSPRES = CWSPRES $CLSS A
758           TCURSYM = CWSPRFS $MASK= 077
759   GOTO    TCURSYM=046, X, TAND  $$ '.OR'./
760   GOTO    STATE=1, X, GTST2, BACKUP, GTST2
761   CALC    CRTOKEN = 1     $$ CURRENT TOKEN = '.OR'.
762   GOTO    EXITO

763   ENTRY   TAND
764   GOTO    TCURSYM=047, X, TLPAR
765   GOTO    STATE=1, X, GTST2, BACKUP, GTST2
766   CALC    CRTOKEN = 2     $$ CURRENT TOKEN = '.AND'.
767   GOTO    EXITO

768   ENTRY   TLPAR
769   GOTO    TCURSYM=051, X, TRPAR
770   GOTO    STATE=1, X, GTST2, BACKUP, GTST2
771   CALC    CRTOKEN = 5     $$ CURRENT TOKEN = '.('.
772   GOTO    EXITO

773   ENTRY   TRPAR
774   GOTO    TCURSYM=052, X, TSCOLN        $$ '.)'/
775   GOTO    STATE=1, X, GTST2, BACKUP, GTST2
776   CALC    CRTOKEN = 6     $$ CURRENT TOKEN = '.)'.
777   GOTO    EXITO

778   ENTRY   TSCOLN
779   GOTO    TCURSYM=077, X, TBLANK
780   GOTO    STATE=1, X, GTST2, BACKUP, GTST2
781   CALC    CRTOKEN = 4     $$ CURRENT TOKEN = '.;'.
782   GOTO    EXITO

783   ENTRY   TBLANK
784   GOTO    TCURSYM=056, X, TUCASE
785   GOTO    STATE=1, LPINWRD, GTST2, STORESW, GTST2

786   ENTRY   TUCASE
787   GOTO    TCURSYM=070, X, TNEGSYM  $$ UPPER CASE'/
788   GOTO    STATE=3, GTST2, X
789   CALC    STATE = STATE + 1
790   GOTO    STORESW

791   ENTRY   TNEGSYM
792   GOTO    TCURSYM=043, X, NRESERV  $$ '+'7'+'/
793   GOTO    STATE=1, GTST2, X, STORESW, NGST3
794   CALC    CRTOKEN = 7     $$ CURRENT TOKEN = '+'7'.
795           CPSW = 1   $$ BACK UP CHAR POS OF SCH WRD
796           STATE = 0
797   GOTO    EXITO
```

GETNSYM

TAND

TLPAR

TRPAR

TSCOLN

TBLANK

TUCASE

TNEGSYM

NGST3

```
        LESSON BUILDER          AT      08:59:28.      ON      04/24/73

798  ENTRY   NRSTR
799  MOVE    BLANK, 10, SEARCHW(WPSW), CPSW-1
800  CALC    STATE = 1
801  GOTO    BACKUP1

802  ENTRY   BACKUP                                                    BACKUP
803  CALC    STATE = 0

804  ENTRY   BACKUP1                                                   BACKUP1
805  CALC    CHTOKEN = -WPSW      $$ CURRENT TOKEN = SEARCH WORD
806          WPSW = WPSW + 1
807          CPSP = CPSP-1
808          CWSPRES = CWSPRES $CLS$ $A
809          CPSW = 1
810  GOTO    WPSW>LWPSW, X, EXITO
811  WRITE   *ERROR*, *TOO MANY KEYWORDS, *PRESS -*N*E*X*T-.
812  GOTO    EREXIT

813  ENTRY   NRESERV                                                   NRESERV
814  GOTO    STATE-1, X, GTST2, STORESW, GTST2

815  ENTRY   GTST2                                                     GTST2
816  CALC    STATE = 2

817  ENTRY   STORESW                                                   STORESW
818  MOVE    TCURSYM,10,SEARCHW(WPSW),CPSW
819  CALC    CPSW = CPSW + 1                                           B
820  UNLOADC SEARCHW(WPSW), SPSCHWD+2*WPSW+2099, 2
821  GOTO    CPSW>LSW, X, LPINWRD
822  WRITE   *ERROR*, *TOO LONG KEYWORD, *PRESS -*N*E*X*T-.
823  GOTO    EREXIT

824  ENTRY   LONGSP                                                    LONGSP
825  WRITE   *ERROR*, *TOO LONG SEARCH PRESCRIPTION, *PRESS -*N*E*X*T-.
                                                                       EREXIT
826  ENTRY   EREXIT
827  CALC    ECLFXI = -1
828  EXIT

                                                                       EXITO
829  ENTRY   EXITO
830  CALC    ECLFXI = 0
831  UNLOADC SEARCHW(1),SPSCHWD+3001,40
832  EXIT    1

------------------------------------------------------------------------------ EDITORS
834  *

                                                                                EDTLSP
835  UNIT    EDTLSP
836  CALC    J = 0

                                                                       EPINIT
837  ENTRY   EPINIT
838  LOADC   POSTFIX(1), SPTFIX+3001, LTSRNGE*LPSTFIX
839  CALC    TEMP = TSRANGE(-PSTACK(1)-LWPSW)
840          X = 405
```

```
        LESSON OUTLOG        AT     08.59.28.    ON     04/26/73

841   GOTO    TEMP=0,NOLESSN,X
842   AT      103
843   WHITE   *L*E*S*S*O*N *N*A*M*E
844   AT      325
845   WRITE   *A*M*S*T*R*A*C*T
846   BREAK

847   ENTRY   NEXTBIT                                              NEXTBIT
848   CALC    J = 101
849   GOTO    JUNICATLG, XEDLSP, X
850   CALC    TEMP = TEMP $CLS$ 1
851           TEMP2 = TEMP $MASK$ 01
852   GOTO    TEMP2=1,X*NEXTBIT
853   BREAK
854   CALC    K = K+100
855   GOTO    K>30000, X, SHWDATA
856   PAUSE
857   ERASE
858   GOTO    EPINIT

859   ENTRY   SHWDATA                                              SHWDATA
860   AT      K
861   LOADC   LESSNM1(I),SUC1,LC1
862   SHOWA   LESSNM1(J)
863   AT      K,12
864   SHOWA   ABSTRCT(J),LABSTRC
865   GOTO    NEXTBIT

866   ENTRY   NOLESSN                                              NOLESSN
867   AT      1005
868   WRITE   *SORRY, BUT NO LESSONS SATISFY YOUR SPECIFICATION.  **
869           ASK LOOSE THE SPECIFICATION AND TRY AGAIN.

870   ENTRY   XEDLSP                                               XEDLSP
871   EXIT    1
872   *
873   *
874   *

875   UNIT    EDTDCO                                                       EDTDCO
876   GOTO    I=0,ENDCORS,X
877   AT      304
878   WRITE   *L*E*S*S*O*N          *T*Y*P*E    *P*E*R*F*O*R*M *E*X*P*C*T*D
879   AT      341
880   WRITE   *T*I*M*E  *R*E*C*O*R*D      *D*A*T*E
881   LOADC   COURCEN(I),SUCO,LCDIRCT
882   CALC    SOR = SCOUTLN*(PCOUTLN(I)-1)*WPLC
883           NLSPEC = LCOUTLN(I)
884           NWLOAD = LCOUTLN(I)*WPLC
885   JOIN    EDTDATA
886   EXIT    1

887   ENTRY   ENDCORS                                              ENDCORS
888   AT      1005
```

```
LESSON BUILDER          AT      08.59.28.     ON      06/24/73

889   WRITE   *THE COURSE IS NOT IMPLEMENTED IN OUR DATA BASE.
890   FAIL    1
891   *                                                                    EDTDSR

892   UNIT    EDTDSR
893   AT      304
894   WRITE   *L*E*S*S*O*N          *T*Y*P*E     *P*E*R*F*U*R*M*A*N*C*E
895   AT      341
896   WRITE   *T*I*M*E  *S*P*E*N*T      *D*A*T*E
897   LOADC   SDC$FCN(1),SUB,NWLOAD
898   CALC    SDA = SSRECORD+(PSRECORD(1)-1)*WPLC
899           NWLOAD = NLSREC*WPLC
900   JOIN    EDTDATA
901   EXIT    1
902   *                                                                    EDTDATA

903   UNIT    EDTDATA
904   CALC    J = 0
905           K = 402
906   AT      3005                                                         NEXTLSN

907   ENTRY   NEXTLSN
908   BREAK
909   *
910   LOADC   NC(LC1*1)*SDB*NWLOAD
911   LOADC   LESSNM1(1),SDC1,LC)
912   CALC    J = J+1
913           K = K+100
914   GOTO    J>NLSREC, RETURN, 1
915   AT      K
916   SHOWA   LESSNM1(CLFSSNW(J))
917   AT      K+4A
918   SHOW    MONTH(J)
919   AT      K+49
920   SHOW    DAY(J)
921   AT      K+52
922   SHOW    YEAR(J)
923   AT      K+56
924   WRITE   / /
925   AT      K+35
926   SHOW    TIME(J)
927   WRITE    MIN.
928   AT      K+23
929   SHOW    PERFORM(J)
930   GOTO    NEXTLSN                                                      RETURN

931   ENTRY   RETURN
932   EXIT    1
------------------------------------------------------------------------ EDTDLD
934   *
935   *
936   *
```

```
937  UNIT    EDTOLD                                                    EDTOLD
938  GOTO    I<0.FNOLESN.X
939  LOADC   LESSNM1(1),SDC1,LC1,LCATL82,LKEYWDT
940  LOADC   LTYPE(0), $OLT, LLTYPE
941  CALC    TEMP = ALESSNN(I)
942  AT      504
943  WRITE   *L*F*S*S*O*N *N*A*M*E*:
944  SHOWA   LESSNM2(I)
945  AT      530
946  WRITE   *T*Y*P*E*:
947  SHOWA   LTYPE(ATYPE(I))
948  AT      704
949  WRITE   *A*R*S*T*R*A*C*T*:
950  SHOWA   ABSTRCT(TEMP),LARSTRC
951  AT      1004
952  WRITE   *C*A*T*E*G*O*R*Y*:
953  CALC    TEMP2 = SUBJCAT(TEMP) $CLSS 62
954          J = 0

955  ENTRY   NEXTSJC                                                   NEXTSJC
956  CALC    J = J+1
957  GOTO    J>NSUBJCT, DKEYWRD,X
958  CALC    TEMP2 = TEMP2 $CLS$ NDSUBJC*6
959          TEMP3 = TEMP2 $MASK$ MASK2
960  GOTO    TEMP3=0, DKEYWRD,X
961  CALC    TEMP2 = TEMP3 $CLSS 54
962  SHOWA   TEMP3,1
963  WRITE   .
964  SHOWA   TEMP2,NDSUBJC-1
965  WRITE   .
966  GOTO    NEXTSJC

967  ENTRY   DKEYWRD                                                   DKEYWRD
968  AT      1204
969  WRITE   *K*E*Y*W*O*R*D*S*:
970  AT      1217
971  CALC    TEMP2 = KEYCODE(TEMP)
972          J = 0

973  ENTRY   NEXTKEY                                                   NEXTKEY
974  CALC    TEMP2 = TEMP2 $CLS$ NBKEYCD
975          TEMP3 = TEMP2 $MASK$ MASK6
976  GOTO    TEMP3=0, DRQTIME, X
977  SHOWA   KEYWORD(TEMP3), LKEYWRD
978  CALC    J = J+1
979  GOTO    J>NKEYWRD, DRQTIME, X
980  WRITE
981  . & BLANKS
982  GOTO    NEXTKEY

983  ENTRY   DRQTIME                                                   DRQTIME
984  AT      1604
985  WRITE   *R*E*Q*U*I*R*E*D *T*I*M*E*:
```

```
      LESSON BUILDER          AT     08.59.28.     ON     06/26/73

 986  SHOW    GIVEN(I)
 987  WRITE     MIN.
 988  CALC    TEMP3 = RELATN(I)
 989          J = 0
 990          K = 1004
 991  LOADC   CRELATN(0), SCRELCM, LREL

 992  ENTRY   XTREL                                                    XTREL
 993  CALC    TEMP3 = TEMP2 $CLS$ NBREL
 994          TEMP3 = TEMP2 $MASK$ 017
 995  GOTO    TEMP3=0, XTOLD, X
 996  AT      K
 997  SHOWA   CRELATN(TEMP3)
 998  WRITE   +1
 999  * WRITTEN TWO CHARS INCLUDING A BLANK AFTER +1
1000  CALC    TEMP3 = TEMP2 $CLS$NBLN
1001          TEMP3 = TEMP2 $MASK$ 077
1002          K = K+12
1003  AT      K
1004  SHOWA   LESSNM1(TEMP3)
1005  CALC    K = K+12
1006  AT      K
1007  SHOWA   ABSTRCT(TEMP3), 42
1008  CALC    K = K+200
1009          K = K-24
1010  GOTO    XTREL

1011  ENTRY   XTOLD                                                    XTOLD
1012  PAUSE
1013  EXIT    1

1014  ENTRY   ENOLESN                                                  ENOLESN
1015  AT      1505
1016  WRITE   +THE LESSON SPECIFIED IS NOT IMPLEMENTED IN OUR DATA BASE.
1017  PAUSE
1018  EXIT    1

      ------------------------------------------------------------------------------------------ INITIALIZE
1020  *

1021  UNIT    INPRECD                                                  INPRECD
1022  CALC    PRECONC(1) = 1
1023          PRECONC(2) = -1
1024          PRECONC(3) = -1
1025          PRECONC(4) = 1
1026          PRECONC(5) = -1
1027          PRECONC(6) = 1
1028          PRECONC(7) = -1
1029          PRECONC(8) = 1
1030          PRECONC(9) = 1
1031          PRECONC(10) = -1
1032          PRECONC(11) = 1
1033          PRECONC(12) = -1
1034          PRECONC(13) = 1
```

```
1035            PRECONC(14) = -1
1036            PRECONC(15) = 1
1037            PRECONC(16) = 1
1038            PRECONC(17) = 1
1039            PRECONC(18) = 1
1040            PRECONC(19) = 2
1041            PRECONC(20) = 1
1042            PRECONC(21) = 1
1043            PRECONC(22) = -1
1044            PRECONC(23) = -1
1045            PRECONC(24) = -1
1046            PRECONC(25) = 0
1047            PRECONC(26) = -1
1048            PRECONC(27) = 4
1049            PRECONC(28) = -1
1050            PRECONC(29) = -1
1051            PRECONC(30) = -1
1052            PRECONC(31) = -1
1053            PRECONC(32) = 5
1054            PRECONC(33) = -1
1055            PRECONC(34) = 0
1056            PRECONC(35) = -1
1057            PRECONC(36) = 1
1058            PRECONC(37) = 1
1059            PRECONC(38) = 1
1060            PRECONC(39) = 1
1061            PRECONC(40) = 4
1062            PRECONC(41) = 1
1063            PRECONC(42) = 7
1064            PRECONC(43) = 1
1065            PRECONC(44) = 1
1066            PRECONC(45) = 1
1067            PRECONC(46) = 1
1068            PRECONC(47) = 8
1069            PRECONC(48) = 1
1070            PRECONC(49) = 9
1071    UNLOADC PRECONC(1),SPPRECD,3001,49
1072    EXIT    1
1073    *
1074    *
1075    *  INITIALIZE LESSON TYPE DECODER

1076    UNIT    ILTDEC                                                  ILTDEC
1077    CALC    LTYPE(0) = '*GENERAL*,
1078            LTYPE(1) = '*EXERCISE*,
1079            LTYPE(2) = '*EXAM*,
1080    UNLOADC LTYPE(0),SOLT,LLTYPE
1081    EXIT    1
1082    *
1083    *  ROUTINE FOR DEBUG

1084    UNIT    DEBUG1                                                  DEBUG1
1085    CALC    LESSNM2(1) = '*MONTECARLO*,
```

```
        LESSON GUIDE        AT     08.59.28.    ON     04/24/73

1086            LESSNM2(2) = *.PL1ARRAY  *.
1087            LESSNM2(3) = *.PL1DATA   *.
1088            LESSNM2(4) = *.PL1NO     *.
1089            LESSNM2(5) = *.PL1IF     *.
1090            LESSNM2(6) = *.PL1TO     *.
1091            LESSNM2(7) = *.PL1LAB    *.
1092            LESSNM2(8) = *.PL1X1     *.
1093            LESSNM2(9) = *.RACETRACK *.
1094            LESSNM2(10) = *.ROOTLAB  *.
1095            LESSNM2(11) = *.SOMAGA   *.
1096            NC(SCATL82+2) = 0851700000000000000000
1097            NC(SCATL82+4) = 0103600000000000000000
1098            NC(SCATL82+6) = 00A7400000000000000000
1099            NC(SCATL82+8) = 0035000000000000000000
1100            NC(SCATL82+10) = 0A7360414414000000000
1101            NC(SCATL82+12) = 0027400000000000000000
1102            NC(SCATL82+14) = 0127400000000000000000
1103            NC(SCATL82+16) = 0116200000000000000000
1104            NC(SCATL82+18) = 0012400000000000000000
1105            NC(SCATL82+20) = 0137400000000000000000
1106            NC(SCATL82+22) = 0046200000000000000000
1107    UNLOADC LESSNM2(1),SUC2,NICTL82=NWLCL87
1108    EXIT    1
1109    *

                                                                                @RELCO
1110    UNIT    @RELCO
1111    CALC    CRELATN(0) = *.NO RELAT  *.
1112            CRELATN(1) = *.PREREQSITE*.
1113            CRELATN(2) = *.SEQUEL    *.
1114    UNLOADC CRELATN(0), SCRELCM= LRFL
1115    EXIT    1

  ------------------------------------------------------------------------------------------------------------------------------------------------------ @EBUG1
1117    *

                                                                                @EBUG3
1118    UNIT    @ERING3
1119    CALC    NC(SKEYWDT+WPKKT+1-2) = *.ARRAY     *.
1120            NC(SKEYWDT+WPKKT+1-1) = *.          *.
1121            NC(SKEYWDT+WPKKT+1) = 0002000000000000000000
1122            NC(SKEYWDT+WPKKT+2-2) = *.ASSIGNMENT*.
1123            NC(SKEYWDT+WPKKT+2-1) = *. STATEMENT*.
1124            NC(SKEYWDT+WPKKT+2) = 0
1125            NC(SKEYWDT+WPKKT+3-2) = *.DATA STRUC*.
1126            NC(SKEYWDT+WPKKT+3-1) = *.TURE      *.
1127            NC(SKEYWDT+WPKKT+3) = 0002000000000000000000
1128            NC(SKEYWDT+WPKKT+4-2) = *.DATA TYPE *.
1129            NC(SKEYWDT+WPKKT+4-1) = *.          *.
1130            NC(SKEYWDT+WPKKT+4) = 0010000000000000000000
1131            NC(SKEYWDT+WPKKT+5-2) = *.DECLARATIO*.
1132            NC(SKEYWDT+WPKKT+5-1) = *.N          *.
1133            NC(SKEYWDT+WPKKT+5) = 0010000000000000000000
1134            NC(SKEYWDT+WPKKT+6-2) = *.FORTRAN    *.
1135            NC(SKEYWDT+WPKKT+6-1) = *.           *.
```

```
1136          NC(SKEYWDT.WPKKT06) = 0
1137          NC(SKEYWDT.WPKKT07-2) = *.GAME      *.
1138          NC(SKEYWDT.WPKKT07-1) = *.          *.
1139          NC(SKEYWDT.WPKKT07) = 044000000000000000000
1140          NC(SKEYWDT.WPKKT08-2) = *.INPUT     *.
1141          NC(SKEYWDT.WPKKT08-1) = *.          *.
1142          NC(SKEYWDT.WPKKT08) = 070000000000000000000
1143          NC(SKEYWDT.WPKKT09-2) = *.ITERATION *.
1144          NC(SKEYWDT.WPKKT09-1) = *.          *.
1145          NC(SKEYWDT.WPKKT09) = 01020000000000000000
1146          NC(SKEYWDT.WPKKT10-2) = *.LABEL     *.
1147          NC(SKEYWDT.WPKKT10-1) = *.          *.
1148          NC(SKEYWDT.WPKKT10) = 0
1149          NC(SKEYWDT.WPKKT11-2) = *.MANAGEMENT*.
1150          NC(SKEYWDT.WPKKT11-1) = *.          *.
1151          NC(SKEYWDT.WPKKT11) = 004000000000000000000
1152          NC(SKEYWDT.WPKKT12-2) = *.NUMERICAL *.
1153          NC(SKEYWDT.WPKKT12-1) = *.METHOD    *.
1154          NC(SKEYWDT.WPKKT12) = 010200000000000000000
1155          NC(SKEYWDT.WPKKT13-2) = *.OUTPUT    *.
1156          NC(SKEYWDT.WPKKT13-1) = *.          *.
1157          NC(SKEYWDT.WPKKT13) = 020000000000000000000
1158          NC(SKEYWDT.WPKKT14-2) = *.PL/1      *.
1159          NC(SKEYWDT.WPKKT14-1) = *.          *.
1160          NC(SKEYWDT.WPKKT14) = 031740000000000000000
1161          NC(SKEYWDT.WPKKT15-2) = *.PROGRAMMIN*.
1162          NC(SKEYWDT.WPKKT15-1) = *.R LANGUAGE*.
1163          NC(SKEYWDT.WPKKT15) = 035740000000000000000
1164          NC(SKEYWDT.WPKKT16-2) = *.PROGRAM FL*.
1165          NC(SKEYWDT.WPKKT16-1) = *.OW CONTROL*.
1166          NC(SKEYWDT.WPKKT16) = 010400000000000000000
1167          NC(SKEYWDT.WPKKT17-2) = *.PRACTICE  *.
1168          NC(SKEYWDT.WPKKT17-1) = *.          *.
1169          NC(SKEYWDT.WPKKT17) = 002040000000000000000
1170          NC(SKEYWDT.WPKKT18-2) = *.REGURTION *.
1171          NC(SKEYWDT.WPKKT18-1) = *.          *.
1172          NC(SKEYWDT.WPKKT18) = 0
1173          NC(SKEYWDT.WPKKT19-2) = *.ROOT FINDI*.
1174          NC(SKEYWDT.WPKKT19-1) = *.NG        *.
1175          NC(SKEYWDT.WPKKT19) = 000020000000000000000
1176          NC(SKEYWDT.WPKKT20-2) = *.SIMULATION*.
1177          NC(SKEYWDT.WPKKT20-1) = *.          *.
1178          NC(SKEYWDT.WPKKT20) = 046000000000000000000
1179  UNLOADC KEYWORD(1).SOKY.LKEYWDT
1180  EXIT    1
```

```
1182  *                                                                    DEBUG2

                                                                           DWGCL01
1183  UNIT    DWGCL01
1184  CALC    LESSON(1) = *.RACETRACK *.
1185          LESSON(2) = *.PL1TO     *.
1186          LESSON(3) = *.PL1NO     *.
1187          LESSON(4) = *.SONAGA    *.
```

```
LESSON GUIDED          AT     08.59.28.     ON     06/24/73

1188                LESSNM1(5) = +.MONTECARLO+.
1189                LESSNM1(6) = +.PL1DATA   +.
1190                LESSNM1(7) = +.PL1F      +.
1191                LESSNM1(8) = +.PL1ARRAY  +.
1192                LESSNM1(9) = +.PL1XI     +.
1193                LESSNM1(10) = +.PL1LAB    +.
1194                LESSNM1(11) = +.ROOTLAB   +.
1195  PACK          TEMP.ABSTRCT(1)=+SIMULATION +EXPERIMENT
1196  PACK          TEMP.ABSTRCT(2)=+P+L/1 +INPUT/+OUTPUT
1197  PACK          TEMP.ABSTRCT(3)=+INTRODUCTION TO +P+L/1 +D+O-STATEMENTS
1198  PACK          TEMP.ABSTRCT(4)=+SOFTWARE +MANAGEMENT +GAME TO TEACH PROGRAMMING
1199  PACK          TEMP.ABSTRCT(5)=+CALCULATION OF AREA BY +MONTE +CARLO METHOD
1200  PACK          TEMP.ABSTRCT(6)=+BEGINNING COMPUTER SCIENCE LESSON
1201  PACK          TEMP.ABSTRCT(7)=+P+L/1 +I+F-+T+H+E+N-+E+L+S+E STATEMENTS
1202  PACK          TEMP.ABSTRCT(8)=+INTRODUCTION TO +P+L/1 ARRAYS
1203  PACK          TEMP.ABSTRCT(9)=+THE EXAM COVERING THE MATERIAL IN PL1 SEQUENCE
1204  PACK          TEMP.ABSTRCT(10)=A LAB ALLOWING STUDENT TO WRITE SIMPLE PL1 PROG
1205  PACK          TEMP.ABSTRCT(11)=+LESSON TO WORK WITH FOUR ROOT FINDING METHODS
1206  CALC          KEYCODE(1) = 0n5003400000000000000
1207                KEYCODE(2) = 0n3404015000000000000
1208                KEYCODE(3) = 0n3404417000000000000
1209                KEYCODE(4) = 0n5005420016000000000
1210                KEYCODE(5) = 0n5004000000000000000
1211                KEYCODE(6) = 0n3402005040000000000
1212                KEYCODE(7) = 0n3407400000000000000
1213                KEYCODE(8) = 0n3401401022000000000
1214                KEYCODE(9) = 0n3410000000000000000
1215                KEYCODE(10) = 0n341040000000000000n
1216                KEYCODE(11) = 0n3011421000000000000
1217  UNLOADC LESSNM1(1),$DC1,LC1
1218  EXIT    1
1219  *
1220  *  INITIALIZE COURSE DIRECTORY FOR DEBUG

1221  UNIT    DEBUG6                                                    DEBUG6
1222  CALC    COURCEN(1) = +.CS101E1   +.
1223                COURCEN(2) = +.CS101N1   +.
1224                COURCEN(3) = +.CS102B1   +.
1225                COURCEN(4) = +.CS103M1   +.
1226                COURCEN(5) = +.CS104K1   +.
1227                COURCEN(6) = +.CS105A1   +.
1228                COURCEN(7) = +.CS105D1   +.
1229                COURCEN(8) = +.CS106F1   +.
1230                COURCEN(9) = +.CS107P1   +.
1231                COURCEN(10) = +.CS108H1   +.
1232                NC(3*1+SCDIRCT) = 0061413130000010n5005
1233                NC(3*2+SCDIRCT) = 0000000000000006010010
1234                NC(3*3+SCDIRCT) = 0000000000000160n3003
1235                NC(3*4+SCDIRCT) = 0000000000000210n4004
1236                NC(3*5+SCDIRCT) = 0000000000000250n6006
1237                NC(3*6+SCDIRCT) = 0000113130000330n5005
1238                NC(3*7+SCDIRCT) = 0000000000000400n4004
1239                NC(3*8+SCDIRCT) = 0000000000000440n3003
1240                NC(3*9+SCDIRCT) = 0000000000000470n5005
```

BEST COPY AVAILABLE

```
        LESSON GUIDED          AT     08.59.28.     ON      04/24/73

1241            NC(3010,SCDIRCT) = 0000000000000540030C3
1242    UNLOADC COURCEN(1),SDCD,LCDIRCT
1243    EXIT    1
1244    *
                                                                                                                    DEBUGS
1245    UNIT    DEBUGS
1246    BREAK
1247    LOADC   SEARCHW(1),SPSCHWD+3001,109
1248    SHOW    POSTFIX(1)
                                                                                                                    DEBUG2
1249    UNIT    DEBUG2
1250    LOADC   SEARCHW(1),SPSCHWD+3001,40
1251    SHOWA   SEARCHW(1)
1252    SHOWA   NC(201,SPSCHWD)
                                                                                                                    DUMMY
1253    UNIT    DUMMY
  ------------------------------------------------------------------------------------------------------------------ DEBUG3
1255    *
                                                                                                                    DEBUGT
1256    UNIT    DEBUGT
1257    CALC    SOCSECN(1) = +.031230729 +.
1258            SOCSECN(2) = +.216338201 +.
1259            SOCSECN(3) = +.216338834 +.
1260            SOCSECN(4) = +.346520078 +.
1261            SOCSECN(5) = +.573443792 +.
1262            SOCSECN(6) = +.100250957 +.
1263            SOCSECN(7) = +.101428469 +.
1264            SOCSECN(8) = +.216338201 +.
1265            SOCSECN(9) = +.352074591 +.
1266            SOCSECN(10) = +.352075612 +.
1267            SOCSECN(11) = +.471930062 +.
1268            SOCSECN(12) = +.763496623 +.
1269            SOCSECN(13) = +.882438257 +.
1270            SOCSECN(14) = +.014057223 +.
1271            SOCSECN(15) = +.655328344 +.
1272            SOCSECN(16) = +.674912645 +.
1273    PACK    TEMP.SNAME(1),DESTON JAMES
1274    PACK    TEMP.SNAME(2),BERGMAN INGRID
1275    PACK    TEMP.SNAME(3),EINSTEIN ALBERT
1276    PACK    TEMP.SNAME(4),FONDA JANE
1277    PACK    TEMP.SNAME(5),ROULEZ PIERRE
1278    CALC    NC(WPS*4) = NC(WPS*4) $MASK$ MASKE $UNION$ 01
1279            NC(WPS*5) = NC(WPS*5) $MASK$ MASKE $UNION$ 013
1280    UNLOADC SOCSECN(1),SSDIRCT,NSDIRCT
1281    EXIT    1
1282    *
                                                                                                                    DUGSCO
1283    UNIT    DUGSCO
1284    CALC    NC(540+1) = 0043600000000227111?4
```

```
        LESSON BUILDER          AT      08.59.28.     ON      06/24/73

1285            NC(SAO+2) = 00550000000000011113A
1286            NC(SAO+3) = 00643000000002151116A0
1287            NC(SAO+4) = 0075000000000000111172
1288            NC(SAO+5) = 0107400000000000111212
1289            NC(SAO+6) = 0112400000000000111224
1290            NC(SAO+7) = 0123100000000000111244
1291            NC(SAO+8) = 0131700000000000111255
1292            NC(SAO+9) = 0015000000000017111270
1293            NC(SAO+10) = 00250000000000201113O1
1294            NC(SAO+11) = 0034300000000024111312
1295  UNLOADC   NC(SAO+1)+SCOUTLN+11
1296  EXIT      1
1297  *


1298  UNIT      DBGSSR                                                    DBGSSR
1299  CALC      NC(1) = 00136000000002271147A
1300            NC(2) = 00243000000002151150S
1301            NC(3) = 00350000000003011151?
1302            NC(4) = 0042400000000143?1524
1303            NC(5) = 00555000000000111545
1304            NC(6) = 0064[illegible]0000024111552
1305            NC(7) = 00750000000000111562
1306            NC(8) = 01000000000000000000
1307            NC(9) = 01100000000000000000
1308            NC(10) = 01200000000000000000
1309            NC(11) = 013000[illegible]0000000000
1310  UNLOADC   NC(1)+SSRECRD+11
1311  *  DEBUG ROUTINE


1312  UNIT      DEBUG5                                                    DEBUG5
1313  AT        3005
1314  WRITE     SEARCH RANGE VECTOR =
1315  LOADC     POSTFIX(1), SPTFIX+3001, LTSRNGE+LPSTFIX
1316  SHOWO     TSRANGE(-PSTACK(1)-LWPSW)
1317  *  CODE FOR DEBUG


1318  UNIT      DEBUG4                                                    DEBUG4
1319  AT        1505
1320  WRITE     POSTFIX +1
1321  DO        DEBUG5, I=1,20
1322  AT        1805
1323  WRITE     SEARCH WORD TABLE +1
1324  DO        DEBUG2, I=1,10
1325  EXIT      1
```

```
LESSON GUIDE        AT     08.59.28.    ON     06/24/73

UNIT      BLOCK      UNIT LOCATION      REFERENCES TO UNIT

AND       SLPRESC          502          498
BACKUP    LEXI             802          760     765     770     775     780
BACKUP1   LEXI             804          801
BUB       SLPRESC          457          452     452
BSEARCH   SEARCH           588          338     420     463     477
CALCSRV   SLPRESC          447          262
CLSCMND   PARSER           734          672
CMDT      SEARCH           599          638
CSYTEM    SEARCH           590
DBGCLG1   DEBUG2          1183          199
DBGSCO    DEBUG3          1283          200
DBGSSR    DEBUG3          1298          201
DEBUG5    DEBUG2          1245         1321
DEBUG1    INITIALIZE      1084          206
DEBUG2    DEBUG2          1249         1324
DEBUG3    DEBUG1          1118          208
DEBUG4    DEBUG3          1318          256
DEBUG5    DEBUG3          1312          266
DEBUG6    DEBUG2          1221          207
DEBUG7    DEBUG3          1256          209
DECRJ     SLPRESC          504          501
DEFINE1   DEFINE1            9
DKEXIT    IDECODE          316          312
DKEYTBL   IDECODE          300          240     279     483
DKEYWRD   EDTOLD           967          957     960
DRQTIME   EDTOLD           983          976     979
DUMMY     DEBUG2          1253
DUMMY1    PARSER           738
EDTDATA   EDITORS          903          885     900
EDTDCO    EDITORS          875          367
EDTOLD    EDTOLD           937          341
EDTOSR    EDITORS          892          433
EDTLSP    EDITORS          835          272
ELLGINE   SLPRESC          535          452
ENDCORE   EDITORS          887          876
ENDFILE   SEARCH           640
ENDLESN   EDTOLD          1014          938
EPINIT    EDITORS          837          858
EPSTOVF   SLPRESC          522          454
EREXIT    LEXI             826          812     823
ERNOCHR   CONTROL          438          385
ERR+/*/               NOT FOUND         232
ETSRUVF   SLPRESC          526          497
EXITO     LEXI             829          762     767     772     777     782     797     810
GETNSYM   LEXI             756          751
GRELCD    INITIALIZE      1110          203
GTST2     LEXI             815          760     760     765     765     770     770     775
                                        775     780     780     785     785     788     793
                                        814     814
HSPRESC   IDECODE          278          239
IUCDCO    CONTROL          347          230
IUCOLD    CONTROL          321          230
IUCOSR    CONTROL          373          230
IUCLSP    IDECODE          238          230     490
IUECODE   IDECODE          215          211     275     343     370     436     442
```

LESSON BUILDER AT 08:59:28. ON 08/24/73

UNIT	BLOCK	UNIT LOCATION	REFERENCES TO UNIT			
ILTDEC	INITIALIZE	1076	204			
INCRP	SLPRESC	449	456	516	538	
INCRW	SLPRESC	494	481			
INDRECD	INITIALIZE	1021	196			
IPLUS1	IDECODE	310	306			
ISRNGE1	SLPRESC	469	458			
ISRNGE2	SLPRESC	491	472			
LEXI	LEXI	746	680			
LONGSP	LEXI	824	743			
LPINWRD	LEXI	747	745	821		
LSRANGE	SLPRESC	539	498			
NEG	SLPRESC	507	731			
NEWTOKE		NOT FOUND				
NEWTOKEN	PARSER	679	852	865		
NEXTBIT	EDITORS	847	661			
NEXTITM	PARSER	655	982			
NEXTKEY	EDTDLD	973	930			
NEXTLSW	EDITORS	907	966			
NEXTSJC	EDTDLD	955	657			
NFOUND	PARSER	663	793			
NGSTJ	LEXI	798	464	478		
NOKEY	SLPRESC	482	841			
NOLESSN	EDITORS	866	792			
NRESERV	LEXI	813	315			
NXTKEYW	IDECODE	304	452	452	467	
OPAND2	SLPRESC	471	251			
PARSER	PARSER	671	914			
RETURN	EDITORS	931	498			
SCALN	SLPRESC	517	462	476		
SETPARW	CALCSRV2	542	855			
SHWDATA	EDITORS	859	518	521	525	
SKEXIT	SLPRESC	529	353	380		
SPCDIRC	CALCSRV2	556	355	382		
SSEARCH	PARSER	649	506			
STACKQ	SLPRESC	509	785	790	793	814
STORESW	LEXI	817	759			
TAND	LEXI	763	779			
TBLANK	LEXI	783	764			
TLPAR	LEXI	768	787			
TNEGSYM	LEXI	791	769			
TRPAR	LEXI	773	774			
TSCALN	LEXI	778	784			
TUCASE	LEXI	786	849			
XEALSP	EDITORS	870	995			
XTALD	EDTDLD	1011	1010			
XTBEL	EDTDLD	992				

```
06/24/73  UNIV OF ILL CY73-24 VER 05.50 06/16/73

UA.59.24.JOH
UA.59.27.ATTACH(TPH)
UA.59.27.ATTACH(TPRINTS)
UA.59.27.LINK.
UA.59.27.MAP.OFF.
UA.59.27.REDUCE.
UA.59.27.TPH.TPRINTS.OUTPUT.
UA.59.35.CP  000.431 SEC.
UA.59.35.PP  007.821 SEC.
```

```
FDPR135  //// END OF LIST ////
```

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-73-587	2.	3. Recipient's Accession No.
4. Title and Subtitle GUIDE-0 -- AN EXPERIMENTAL INFORMATION SYSTEM --			5. Report Date August, 1973 6.
7. Author(s) Shinnichi Murai			8. Performing Organization Rept. No.
9. Performing Organization Name and Address Department of Computer Science University of Illinois at Urbana-Champaign Urbana, Illinois 61801			10. Project/Task/Work Unit No. 11. Contract/Grant No. NSF GJ-31222
12. Sponsoring Organization Name and Address National Science Foundation Washington, D.C.			13. Type of Report & Period Covered 14.

15. Supplementary Notes

16. Abstracts

The experimental information system GUIDE-0 is a bibliographic aid for those students who are taking the introductory computer science courses some of the material of which are implemented as PLATO-IV lessons. The functions, the data base and the detailed description of each module of the system are presented.

17. Key Words and Document Analysis. 17a. Descriptors

information retrieval, information systems
conversational information system
interactive information system
computer-assisted instruction
data base

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
release unlimited	UNCLASSIFIED	101
	20. Security Class (This Page) UNCLASSIFIED	22. Price